SEGA

パーソナルコンピュータ

SC-3000

BASIC LEVEL II

テキスト

セガ・エンタープライゼス

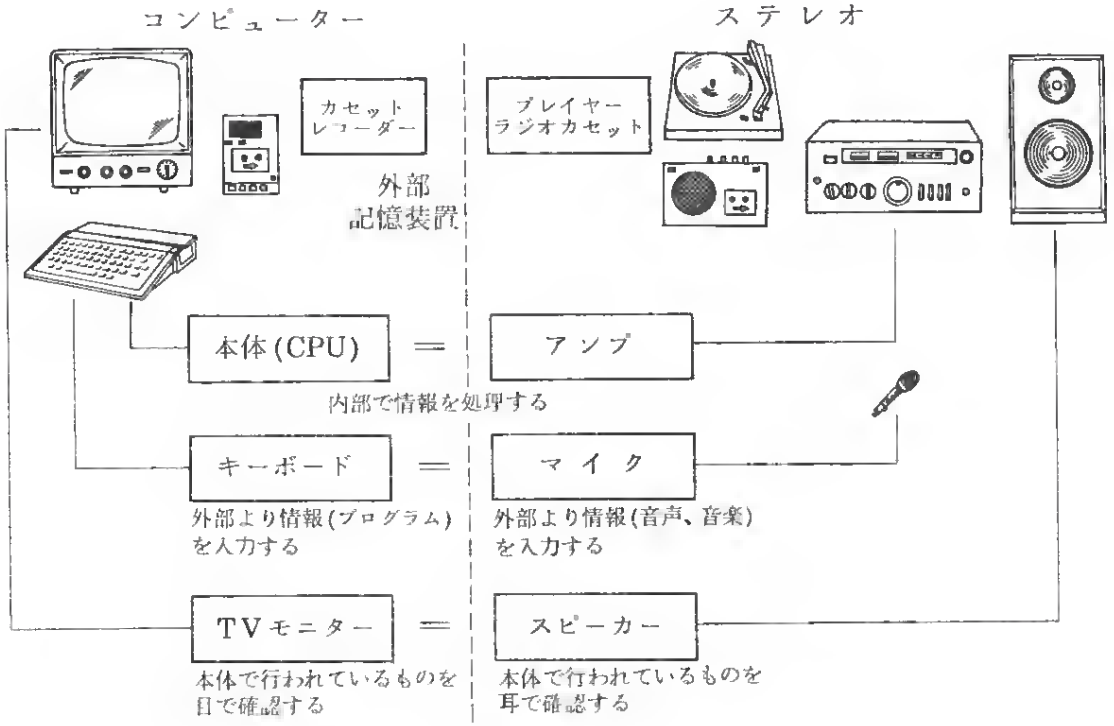
BASIC LEVEL II SEGA

目　　次

はじめに

コンピュータは、もう誰れでも簡単に扱える時代になって来ました。しかし、コンピュータはなんだろうか、言葉を知っていても何をするものか、よく判らない方もおられると思います。

ステレオと比べてみましょう。



図のような仕組になっていて、外部よりプログラムという情報を入れますと、結果がTV画面に映されます。

プログラム用の言語は何種類かありますがここでは、BASIC（ベーシック）言語を使ってみましょう。

コンピュータを扱うには、すでに出来上っているプログラムを使う事も含まれます。しかし、自分専用のプログラムを使いたいのならばプログラム用言語を覚えなければなりません。

コンピュータ用プログラム言語は、用途により沢山開発されております。その中でも、初めてコンピュータを使う人にも理解されやすく、一般的なBASIC言語を使ってプログラムを作ってみましょう。

BASIC言語を使って出来る事は、

1) 計算
2) 文章やデータのファイル
3) 図形や絵を描く
4) ゲームや音楽を楽しむ

その他、色々な事が出来ます。

この便利なコンピュータを自在に扱う為に、BASIC言語をマスターして下さい。 言語と言うと大変難しく聞えますが、命令語の量は決して多くありません。

プログラムを構成する基本的な命令文は、案外少ないのです。まず、基本になる文を10語程覚え、あとは、テキストを見ながら、少しづつ覚えれば、自然に全部覚えてしまいます。

まずテキストを見ながら、キーをたたいて下さい。きっとすぐエラーが出るでしょう。しかし、恐れずにキーをたたいて下さい。

やがて、コンピュータは、あなたの一番忠実な友達になってくれます。

第１章　コンピュータの扱い方

本体の扱い方

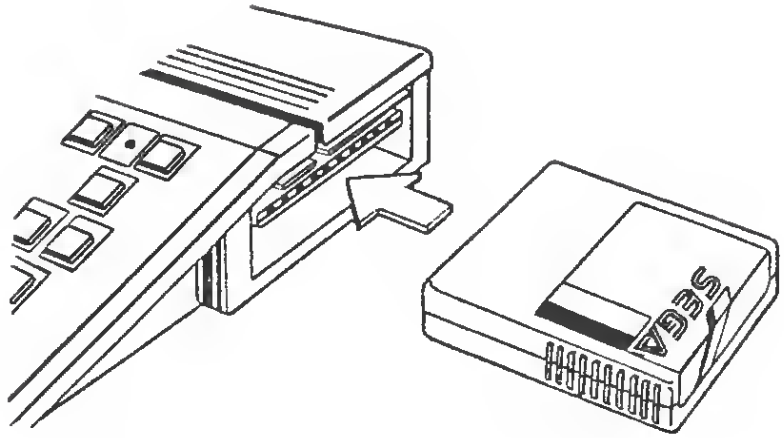
まず、SC－3000本体と一緒に入っている取扱い説明書を読んで下さい。

1. スイッチボックスを、テレビに接続しましたか。
   （ビデオ入力のあるテレビの場合は専用ケーブルで、本体とテレビのビデオ入力端子と音声端子に直接つないで下さい）

2. スイッチボックスをCOMPUTER（コンピュータ）側にして、テレビのチャンネルを１または２チャンネルにして下さい。

3. 本体のチャンネル切替スイッチを、１または２チャンネルの空いた方に合せて下さい。

4. BASICカートリッジを正しく差し込んで下さい。



5. 接続が終ったら、ケーブルの接続を確かめて下さい。正しく接続されていましたら、テレビの電源を ON にして下さい。
次に、コンピュータの電源も ON にして下さい。

SC－3000 に BASIC のカートリッジを挿入しますと、もうコンピュータになりました。ためしにプログラムを入力してみましょう。

5.5

```
10  SCREEN  2,2 [CR]
20  CLS [CR]
30  LINE(80,50)-(180,100), C, BF [CR]
40  C=C+1 [CR]
50  IF  C=16  THEN  C=0  [CR]
60  GOTO  20 [CR]
RUN [CR]
```

（0は一番上の列の0を押して下さい。それが数字の0です。）

1行打ち終ったら [CR] キーを押して下さい。入力したプログラムがコンピュータに記憶されカーソルが 1段下ります。

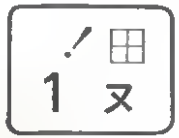
キーの使い方がむずかしいかも知れませんが、一字づつゆっくりと打ってみてください。

キーボードの使い方

キーボードには、英文字、数字、カナ文字、記号の書き込まれたキーがあります。

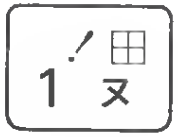
1つのキーには4つの文字あるいは表示があり

例

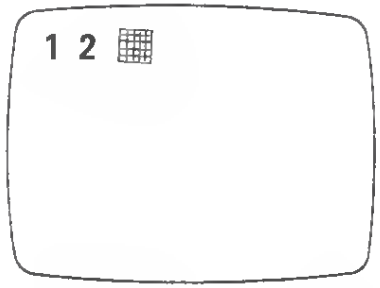


キーの配列と間隔は、英文タイプライターと同じですから、両手で楽に打てます。

まずキーを打ってみましょう。

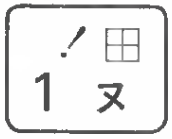


のキーを打って下さい。



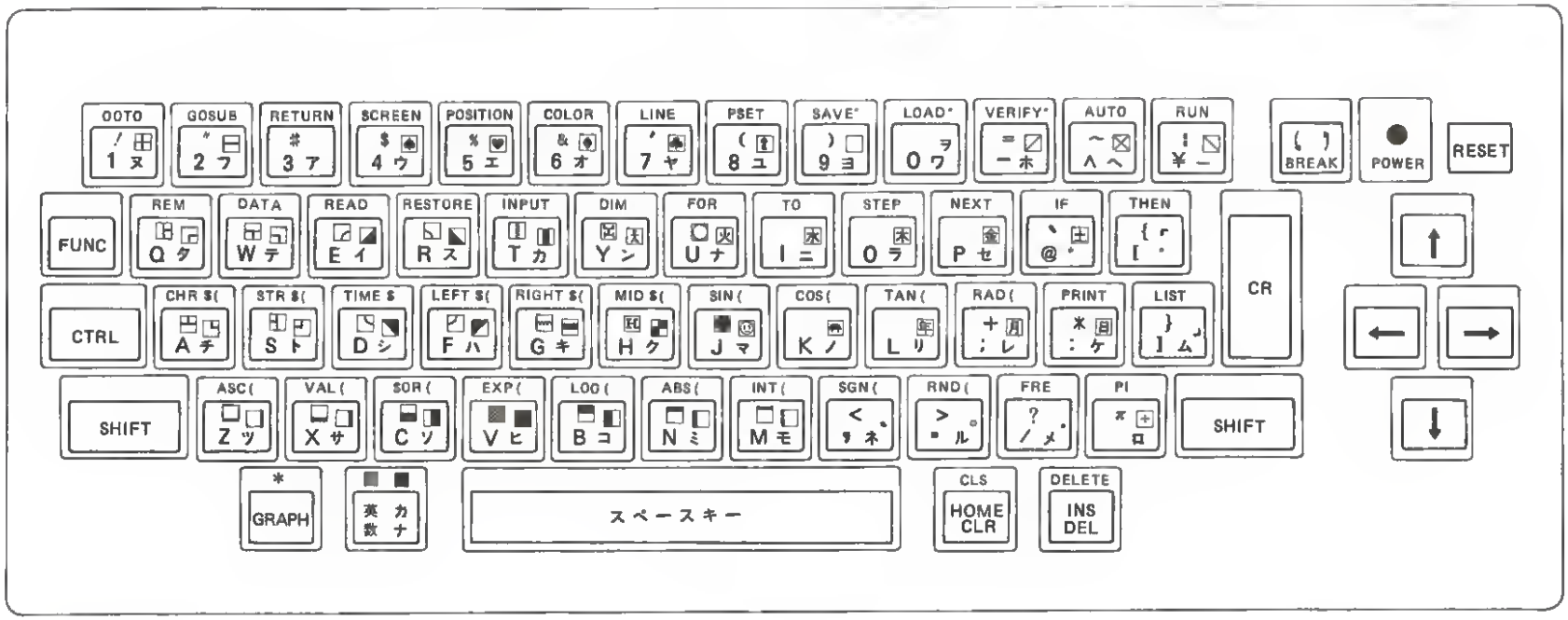
１２と画面に出て来ましたね。そのままキーを打つと、キーに書かれた文字や記号の内で、左下の文字や記号が画面に出ます。

それ以外の文字や記号を画面に出すには、SHIFT（シフト）キーや、GRAPH（グラフィック）キー、カナ キーを使います。

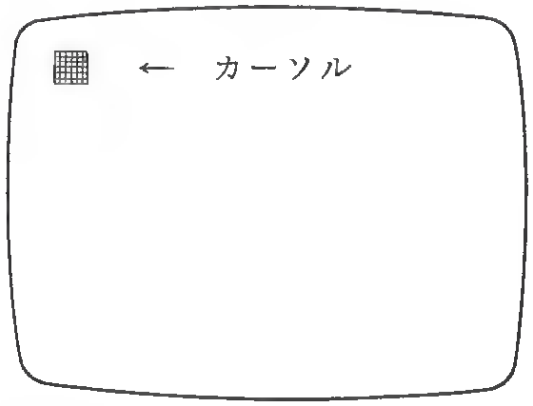
SHIFT キーを押し 

を押すと ！ が画面に出ます。

キーの組合せ


GOTO
GOSUB
RETURN
SCREEN
POSITION
COLOR
LINE
PSET
SAVE"
LOAD"
VERIFY"
AUTO
RUN
BREAK
POWER
RESET
FUNC
REM
DATA
READ
RESTORE
INPUT
DIM
FOR
TO
STEP
NEXT
IF
THEN
CR
CTRL
CHR $(
STR $(
TIME $
LEFT $(
RIGHT $(
MID $(
SIN(
COS(
TAN(
RAD(
PRINT
LIST
SHIFT
ASC(
VAL(
SQR(
EXP(
LOG(
ABS(
INT(
SGN(
RND(
FRE
PI
SHIFT
GRAPH
英数
カナ
スペースキー
CLS
HOME
CLR
DELETE
INS
DEL

カーソル



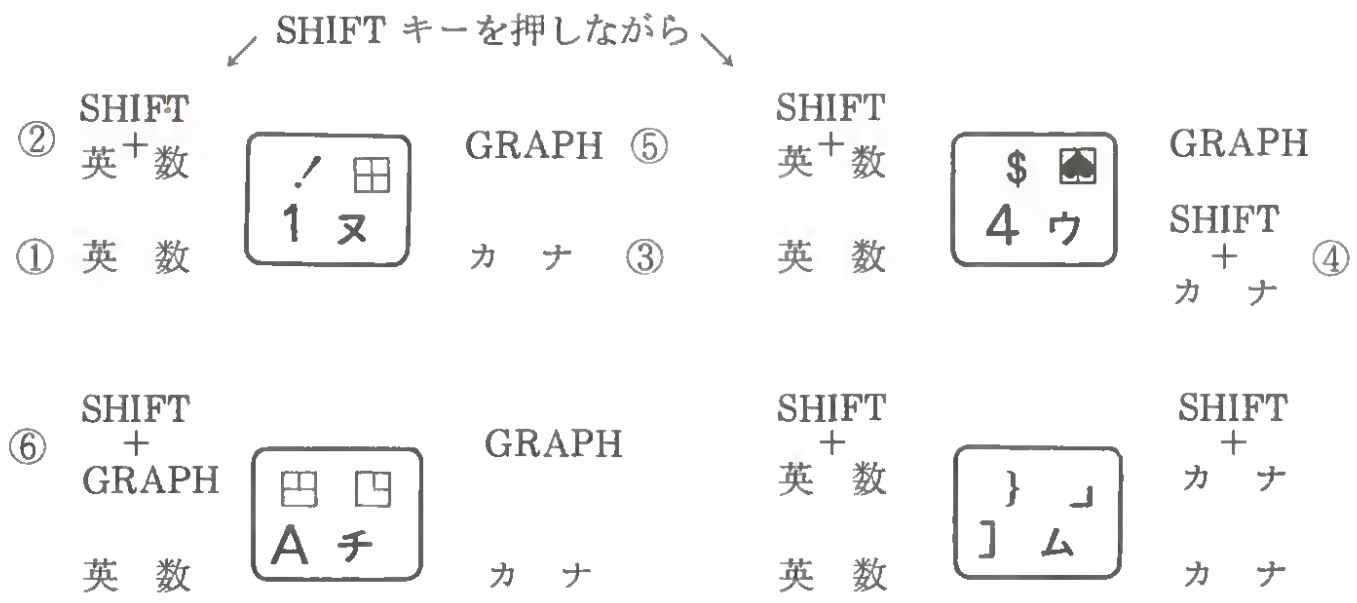
画面左上に ▦ が点滅しています。これは、カーソルと言って、キーからの入力された文字や記号が表示される位置を示しています。
カーソルを移動させるには、キーボード右側の矢印のある、四個のキーで移動させます。

カーソルは、各モードで形が異なります。

英数モード …… ▦
カナモード …… ■
グラフモード …… ＊

キーボード左下にある 英数 カナ 、 GRAPH のキーを押して下さい。カーソルが変ります。元に戻すには、もう一度同じキーを押すと、カーソルは ▦ に戻ります。

キートップに書かれた文字、記号を画面に出すには６つの方法があります。



SHIFT （シフトキー）

左右の SHIFT キーは同じ働きをします。

SHIFT キーを押したまま、数字のあるキーを打つと記号が入力されます。

SHIFT キーを押したまま、英文字のキーを打つと、英文字が小文字で入力されます。

GRAPH （グラフィックキー）

グラフィック記号を入力するためのものです。カーソルは＊に変ります。

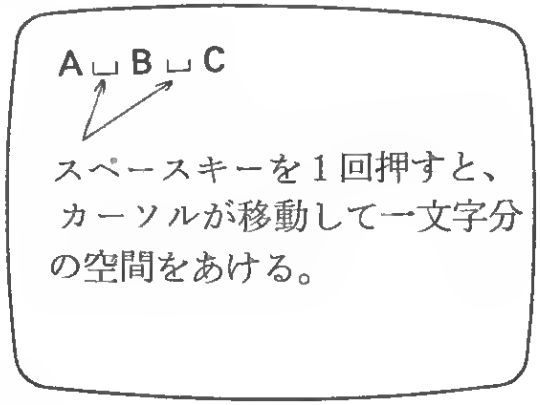
グラフィック記号は、SHIFT キーと併せて使えます。

英数 カナ

カナ文字を入力する場合に使います。カーソルは、■ に変ります。

カナ文字には、アイウエオヤユヨツのように、小さな文字も必要です。その他に、「」 。、・ーの記号も使います。これらの文字や記号は、SHIFT キーと併せて使います。

SPACE （スペースキー：文字や記号の間の空間をあけます。）



一番下段の長いキーは、スペースキーで、文字や記号の間に、スペースを入力します。
コンピュータの場合、空間も文字と同じように一つの情報として扱われます。

今までのキーを使って、色々な文字を入力して下さい。

特殊キー

今までにキーで入力した文字や記号で、画面が一杯になりましたね。必要のない文字や、記号をいつまでも画面に出しておくのは邪魔です。では、すべて消して見ましょう。

使用するキーは

HOME/CLR （ホーム/クリア）

このキーを押すと、画面上の文字が消えて、カーソルが左上のホームポジションに戻ります。
画面上の文字、記号等がすべて不必要な時に使ってください。
キーを打ち始める前に使ってください。
SHIFT キーを押しながら、このキーを押すと、文字は消えずにカーソルだけが、ホームポジションに戻ります。最初の文字、記号等を修正するときに使います。
今までに入力した文字や記号は、プログラムとしてコンピュータの内部には、記憶されていません。
プログラムとして記憶させるには、行番号又は、文番号と呼ばれるものが必要です。

行番号とは、命令文を入れた場所の番地のようなものです。
アドレス（番地）とも言われます。

CR （キャリッジ・リターン）又は（リターン）
コンピューターの場合は、キーで画面に文字を入力しても、メモリーにはまだ記憶されておりません。 CR キーを押して、初めて記憶されます。

何か文字を入力して CR を押して下さい。
Syntax Error と出ましたね。（シンタックスエラー）
コンピュータに入力する場合、一定のきまりで入力しなければなりません。これをコンピュータの文法と言います。文法が間違っているとエラーになります。
文法といっても、国語や、英文法にくらべると、決してむずかしくありません。

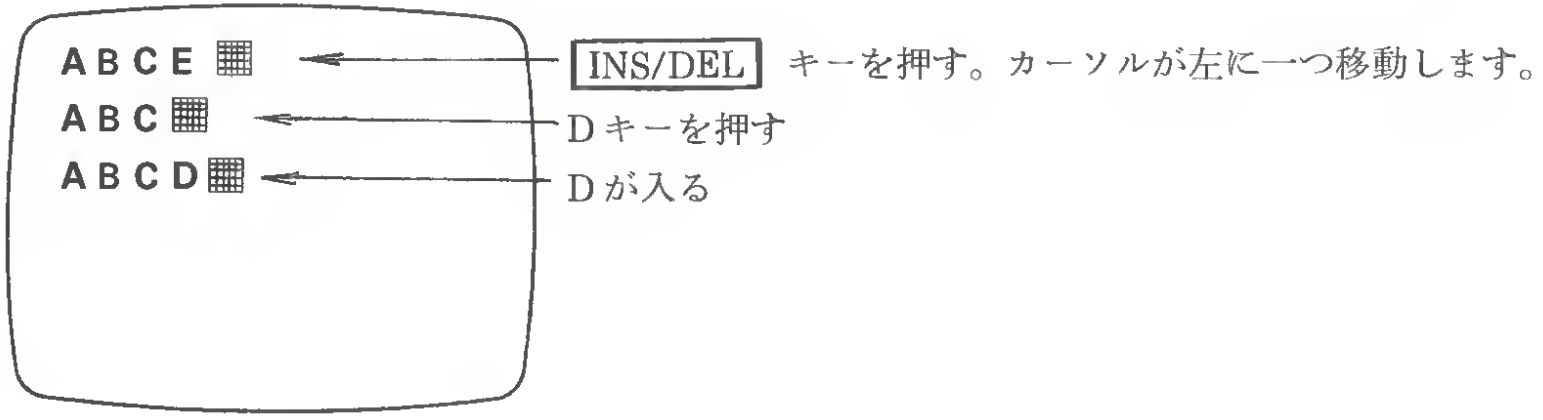
エラーについては、エラーメッセージ表を参照して下さい。

INS/DEL (インサート/デリート)

INS/DEL キーは、文字を一文字づつ消したり追加する場合に使います。

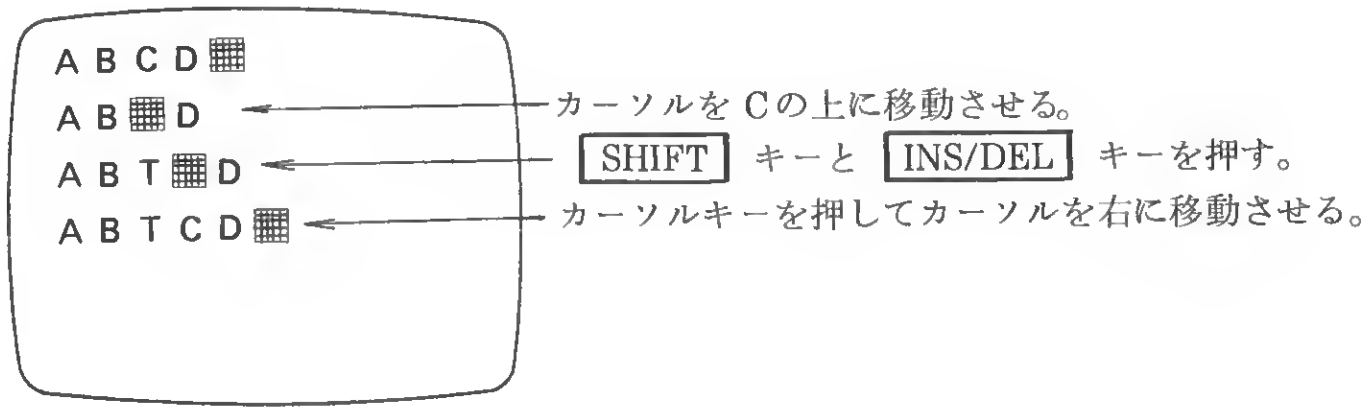
INS (インサート) は、文字を追加する意味です。
DEL (デリート) は、文字を消す意味です。

A B C D と入力する所を、A B C E と打ってしまった場合、INS/DEL キーを押すと、カーソルが一文字分後にさがり、E の文字を消します。そこでもう一度D を入力しますと、A B C D と直りましたね。



こんどは、ＡＢＣＤの間に何か文字を入れてみましょう。
カーソルをＡＢＣＤのＣの上に持って来て下さい。

SHIFT キーを押したまま、 INS/DEL のキーを押して下さい。 カーソルの点滅が早くなりましたね。何か文字を入力して下さい。
Ｂの次に、今入力した文字が入り、ＣＤが右側に一つずれましたね。



インサートモードの時（カーソルが早く点滅している間）は、何文字でも受付けます。

インサートモードから抜け出す場合は、

① CR キーを押す（一般的に良く使うキーです）

② カーソルキーを押す

③ SHIFT ＋ INS/DEL キーを押す

いづれかで、元の状態に戻ります。

プログラムを修正した場合、CR キーを押さないと、メモリー内部は書き替えられません。実際に、キーを操作して確かめて下さい。

FUNC（ファンクション）

いつも使う言語を記憶しているキーです。

キーの上方に GOTO と書いてありますね。それぞれのキーにも英文が書いてあります。これは、BASIC で使われる命令文です。

FUNC キーを押したまま、どれかキーを打って下さい。キーの上方に書かれている命令文が入力されます。

これは、プログラムを作るのに便利です。

[⤵⤴/BREAK] (画面切替 / ブレーク)

プログラムを実行中に、プログラムを停止させたい時に使います。

[SHIFT] キーを押したまま、[⤵⤴/BREAK] キーを押しますと、画面が変ります。これは、このコンピュータが、２つの画面を持っているので切替えて見るためです。

一つの画面はテキスト画面で、プログラムを書く画面です。

もう一つの画面は、グラフィック画面で、グラフィックを表示する画面です。

[⤵⤴/BREAK]

⤵⤴ ➡ 画面を切替える時に使います

[SHIFT] キーを押したまま使って下さい。

これは、コンピュータが、２つの画面を持っているからです。

BREAK ➡ プログラムを実行中にプログラムの実行を停止させたい時に使います。

CTRL (コントロール)

CTRL キーを押したまま、コントロールキーの表にある文字キーを打ちますと、説明にある動作が行われます。

コントロールコード

キー操作	PRINT CHR$(値);	機　　能
CTRL + A	PRINT CHR$(1);	NULL 文字なし
C	——	BREAK プログラムの実行中止
E	5	カーソル以降の文字をクリア
G	7	BELL ピッと音を出す
H	8	DEL 文字をデリート
I	9	HT 水平 TAB
J	10	LF ラインフィード
K	11	HM カーソルを左上端に戻す
L	12	CLR 画面のクリア
M	13	CR キャリッジリターン
N	14	カナ ↔ 英数字切りかえ
O	15	( ) 画面をテキスト ↔ グラフィック切りかえ

キー操作	PRINT CHR$（値）;	機　　能
P	16	標準文字サイズ
Q	17	横2倍文字サイズ (SCREEN 2)
R	18	INS インサート
S	19	キー入力（A～Z）シフトなし大文字
T	20	〃　（a～z）シフトなし小文字
U	21	ラインをクリアしカーソルを左端に戻す。
V	22	ノーマルモード
W	23	GRAPH キー入力グラフモード ↔ 英字切換
X	24	クリック音の ON ↔ OFF 切換
−	28	⇨ カーソル移動
−	29	⇦　　〃
−	30	⇧　　〃
−	31	⇩　　〃

プログラム中でコントロールコードを使う場合は、
PRINT CHR$（値）で入力して下さい。

RESET （リセット）

プログラムを実行中、又は、画面に異常が出た場合、RESET キーを押しますと1 ～2 秒後に、電源を入れた時と同じ状態の画面に戻ります。

このキーを押すと、その時点で行っている処理が中断され、まだ使用していないメモリーの大きさが表示されます。

×××　Bytes　Free

このキーを押しても、入力したプログラムは消えてしまう事はありません。

以上で、キーの説明は終りました。もう一度、キーを使って次のプログラムを打ち込んでみましょう。

415

10　SCREEN　2,2：CLS　CR キー

「線を引く　X座標　Y座標　X座標　Y座標　色番号

20　LINE　（50，50）－（150，150），5　CR キー

90　GOTO　90　CR キー

RUN　CR キー　　RUNはプログラムを実行させる命令です。

こんどは色番号の後に、Bを入れて下さい。

箱を描きます。

```
5 , B    CR
```

色を塗る。

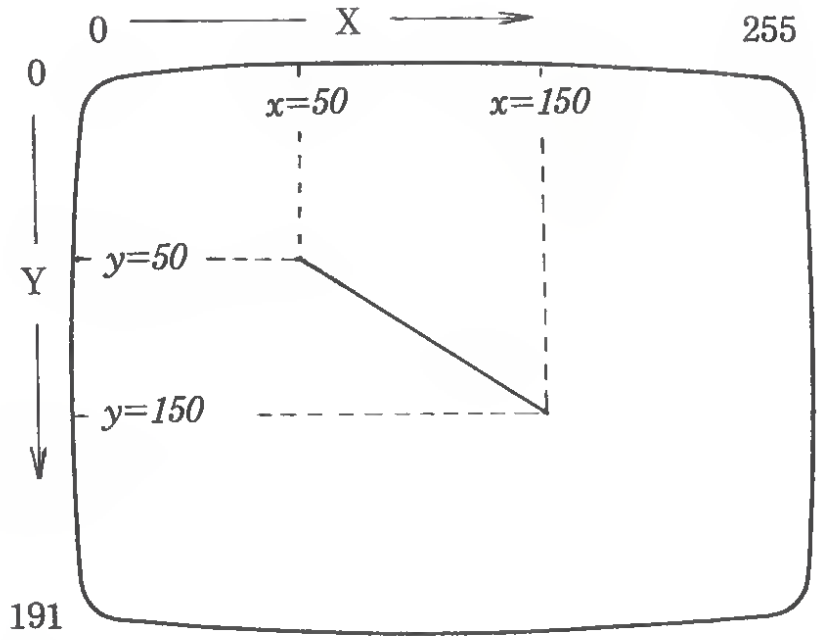
```
5 , B F
```

このように、コントロールコードにより、線や箱や色が描けます。

グラフィックモードの座標

X 軸　0 ～ 255

Y 軸　0 ～ 191



座標の数字を変更すると、線の引き方が変ります。

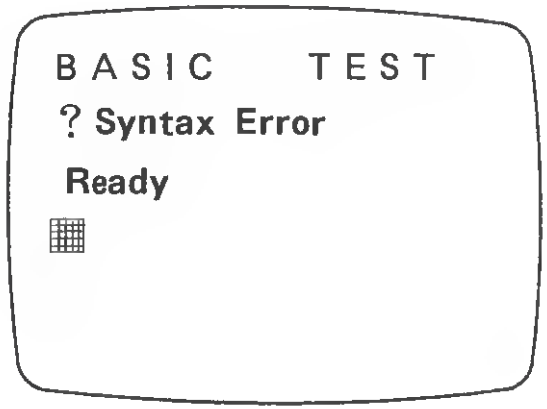
数値や文字を変える場合は、カーソルキーを使って書き替えてから CR キーを押してください。

ダイレクト・モード（直接命令）

コンピュータに直接仕事をさせる方法です。
まず文字を書いてみましょう。

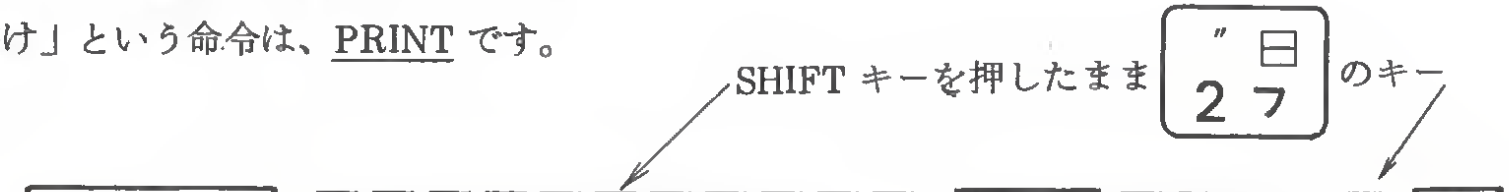
[HOME/CLR] [B][A][S][I][C] [SPACE] [T][E][S][T] [CR]

画面には



と出ます。

エラーとなったのは、コンピュータが入力した命令が理解出来ないからです。
今コンピュータに、BASIC TEST と、書かせたいのですね。その場合は、コンピュータに、入力した文字を「書け」と命令しましょう。
「書け」という命令は、PRINT です。



と 入力して下さい。

PRINT 文は、画面に表示させる命令文です。

画面には、

```
PRINT"BASIC  TEST"
BASIC  TEST
READY
```

キーで入力した文字
PRINT 文で出力した文字

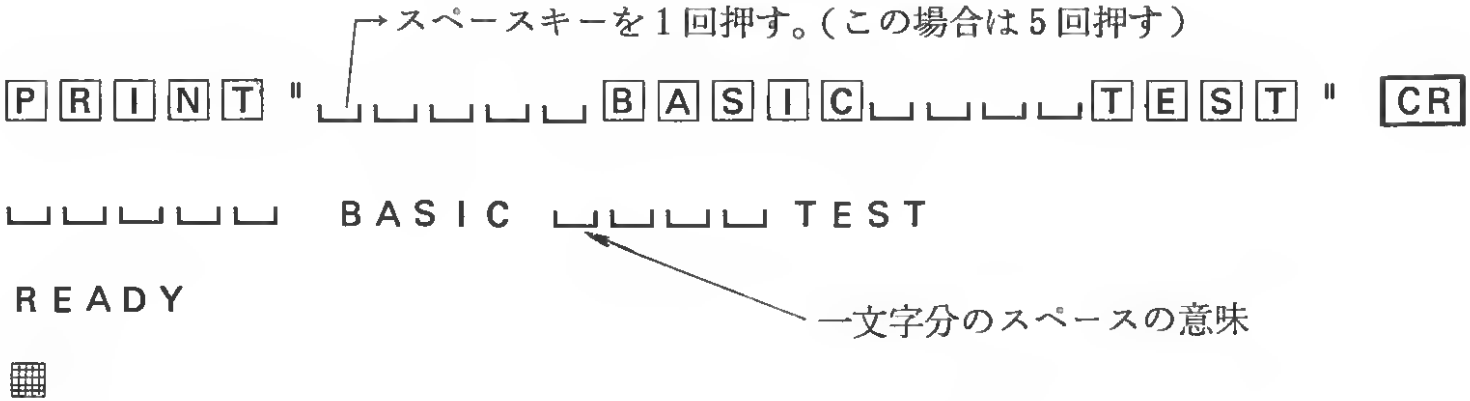
今度は、エラーになりませんね。正しく入力されたからです。

画面に、文字や記号を書きたい時には、PRINT 文を使います。

PRINT 文で、コンピュータに文字や記号を書かせる場合は必らず、"（ダブルクォーテーションマーク）を、文字の両側に付けなければなりません。（又は引用符といいます）
これは、" " で囲んだ文字や記号を、一つの固まりとして扱っているからです。

数式の場合は必要ありません。

今度は、自分の名前を書いて下さい。英文字や、カナ文字を使って下さい。
" " の間には、文字の他、数字や記号、グラフィック記号も入れられます。
" " の中に、空間を置くと文字と同じように扱われます。



このように、PRINT 文を使わないと、コンピュータは入力された結果を見せてくれません。
PRINT 文は、プログラム上で大変よく使われる命令文です。

こんどは、少し違う使い方をします。

[?] [2] [+] [4] [CR]　　?はPRINT の代りに使えます。

6

READY

コンピュータに、２＋４の計算をして、答をPRINTするように命令しました。
計算の結果もPRINT文で表示させます。
[?] の記号は、PRINT文の省略形で、どんなコンピュータでも同じです。

コンピュータに計算をさせる。

四則演算

コンピュータで計算をする場合、計算に使う記号が、普通の計算と違う記号があります。

		呼び名	
たし算	＋	プラス ……	[+]
ひき算	－	マイナス ……	[−]
かけ算	＊	アスタリスク ……	[×]

わり算　／　スラッシュ …… ÷

べき乗　∧　…… X^n

計算に使う記号の事を**演算子**と呼びます。
演算子と違いますが、数式に使う（ ）カッコがあります。
数値（数の大きさ）を比べる、**比較演算子**というのがあります。

以上をまとめたのが次の表です。使う前によく見ておいて下さい。

演算子表

	記号	使用できる場所		内　　容	優先順位
		数値式	文字式		
算術演算	∧	○	×	べき乗　（0∧0は1）	1
	＋	○	×	符号　＋	2
	－	○	×	符号　－	
	＊	○	×	乗算	3
	/	○	×	除算	
	MOD	○	×	余り	4
	＋	○	○	加算　（文字式の場合は文字の結合）	5
	－	○	×	減算	

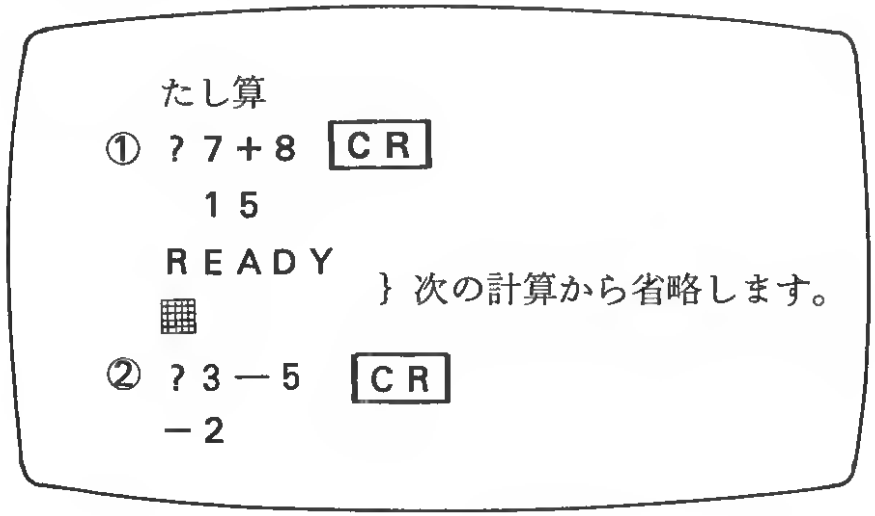
- カッコ（　）が最も優先度が高い。
- 同じ優先順位の演算子が 2 つ以上使われた時は、左側の演算を優先する。
- 加算記号“+”を文字式に用いた時は、結合をあらわす。

（例）“ AB ”＋“ C ” ⇨ “ ABC ”

算術演算は 10 進で行っている。→例えば 0.01 きざみの値でもけたおちがない。
論理演算は 2 進で行っている。
算術演算は 10 進 12 けた計算、 11 けた表示。

キーボードで注意する事があります。キーボードの右下の方に [÷] の記号がありますね。これは、グラフィック記号ですから、計算には使えません。

計算をもう一度やってみましょう。




```
PRINT5＊6
□ 30
```

↑└答が、正の数の場合は、＋記号が省略されるので一文字空く。

```
PRINT10／3
 3．3333333333
```

PRINT は [FUNC] キーと [＊ 日 / ： ケ] を使っても便利です。

```
?3^4 [CR]（3×3×3×3）の意味
 81
```

演算機能は、精度の高い10進計算11桁表示で行っております。

```
?1000000＊10000 [CR]
 100000000000                    11桁

?10／3
 3．3333333333 [CR]    少数点以下10桁
```

これより大きい数字や、小さい数字の場合は、科学的表記法が使われます。

```
?19380000000＊10000000
 1．938E+16 （1．938×10^16）
```

計算の優先順位

```
? 6 + 2 * 4
```

2つの数式が入った計算の場合は、計算は先頭から行われるでしょうか。

四則演算には、優先順位があります。

優先順位は、演算子表の通りです。

もう一度、例題を実行しましょう。

```
? 6 + 2 * 4
  1 4
```

かけ算から先に計算されていますね。もし、たし算を先に計算したい時には、カッコを使います。

```
( 6 + 2 ) * 4
? 3 2
```

1行の計算式の中に、加減乗除を入れる場合は、先に計算したい式にカッコを付けます。カッコ

は、何重にも付けられますが、数学のように大カッコ、中カッコ、小カッコはありません。いつでも小カッコ一種類だけです。

```
   ④      ①    ③    ②
? 3＊（（ 8＋6 ）／（ 4－2 ））
  2 1
```

上の例題の計算は、番号の順番に行われます。計算順位が同じであれば、左側の式から計算されます。

カッコを使う場合、左カッコと右カッコが同数でないとエラーになります。必らずカッコの数は確認して下さい。

PRINT 文のもう一つの使い方

PRINT 文の始めに、＂ ＂で囲んだ文を使いました。これを計算式に入れると、どうなるでしょうか。

```
                    必らず入れてください。
? ＂ 2＋3＝＂ ; 2＋3
2＋3＝   5
```

今までは、答えだけしか表示されませんでしたが、今後は式も表示されましたね。これは、2つの文の組み合せたものです。
"2+3"は、計算式でなく、文字として扱われたので、そのまま表示されます。
;（セミコロン）で区切られた後の文は、2+3の式として扱われましたので、答が表示されました。

もう一つの例

```
?"2+3=",2+3
 2+3=                    5
 READY
```

答が離れた所に出ましたね。(,) コンマを使うと画面の端から20桁目に離れた場所に表示されます。

プリント文を使う時は、セミコロンとコンマを上手に使い分けると、見やすい表示が出来ます。

PRINT 文で、(,) コンマや、(;) セミコロンの使い方

```
PRINT "A";"B";"C";"D";"E";"F"
RUN
A  B  C  D  E  F

PRINT "A","B","C","D","E","F"
A                    B
C                    D
E                    F
```

今まで、色々な計算式を習いましたが、算数で使うイコール (=) 記号が計算式の中に出て来ませんでしたね。コンピュータの場合は、＝記号は別の使い方があります。これは、次の章のプログラムの作り方で説明します。

第３章　プログラムの作り方

BASIC でプログラムを作ってみましょう。キーボードから入力されたプログラムは、コンピュータ内部のメモリーの中に記憶されます。メモリーには、コンピュータが一連の仕事をする為の手順を示す番号を付けます。

この番号を、行番号（文番号またはラインナンバー）と言います。コンピュータは、行番号の小さい側から、大きい側へと順番に行なわれます。

行番号の後には、文章（ステートメント）を書きコンピュータに仕事（実行）させます。

LET，変数

例

```
10 LET A=3                    CR
20 LET B=5                    CR
30 LET C=A+B                  CR
40 PRINT C                    CR
50 END                        CR
```

R U N　　CR

8

READY

RUN という新しい文が出て来ました。これは、コンピュタに仕事を命令する文です。

プログラムを実行させる場合、RUN（ラン）させる、とか、走らせると言います。

LET，変数

LET（代入文）は、変数に数値を入れる命令文です。

```
10  LET  A=3
```

Aは変数という空箱と考えて下さい。この式は、Aが3に等しいではなく、Aという箱に3という数値を入れる事です。
LET文は、省略する事が出来ますので、

```
10  A=3
```

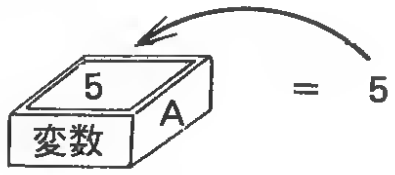
と入力する事が出来ます。

30　LET　C＝A＋B

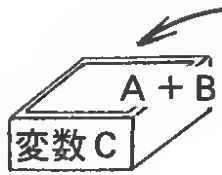
これも、LETを省略出来ます。

イコール(=)の使い方ですが、等しいという意味ではなく、代入文を意味しています。

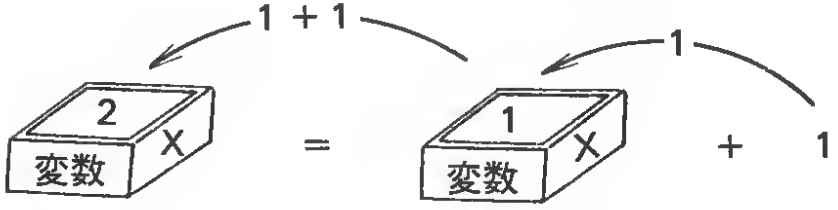
変数の図解



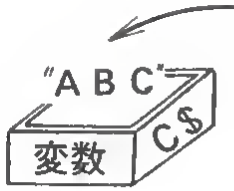
＝　5　　A＝5　　Aに5を入れる



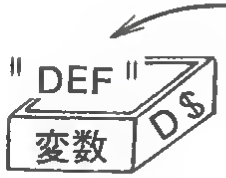
C＝A＋B　　Cに A＋B の結果を入れる。



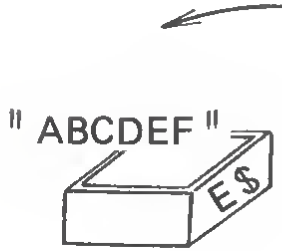
X＝X＋1　　Xに1を加えた結果を又同じ変数Xに入れる。



="ABC"　　C$="ABC"　　C$にABCという文字列を入れる。



="DEF"　　D$="DEF"　　D$にDEFという文字列を入れる。



="ABC"+"DEF"　　E$=C$+D$

E$にABCとDEFのつないだものを入れる。

代入文の使い方で

X＝X＋1

という使い方があります。算数では有り得ない式ですが、代入文では普通に使われます。

例

```
１０　ＣＬＳ
２０　Ｘ＝Ｘ＋１
３０　ＰＲＩＮＴ　Ｘ；
４０　ＧＯＴＯ　２０　　← 行番号
ＲＵＮ
　１　２　３　４　５　６　７　８．．．
```

１文字置きに数字が連続して出て来ます。

Ｘは始めは０です。

$X^{(0)} = X^{(0)} + 1$ なので、左辺のＸに１が代入されます。行番号40から行番号20に飛び、再び $X^{(1)} = X^{(1)} + 1$ に戻りＸは１が加えられて、左辺のＸは２になります。

この、Ｘ＝Ｘ＋１の様な使い方はコンピュータでは多く使われます。

文字変数

文字変数の場合は、$（ダラー）マークを付けます。

```
10  A＝3
20  B＝5
30  C＝A＋B
40  M$＝"コタエ␣"
50  PRINT  M$；C
RUN
コタエ  8
READY
▦
```

←スペースを表わします。

```
40  M$＝"コタエ␣"
```

←スペースキーで1文字空ける。

これは、M$という箱に"コタエ"という文字のかたまりを入れました。
文字を、一つのかたまりとするには、ダブルクォーテーション、(") で左右を囲んで使います。
もし忘れるとエラーになります。
文字変数の場合は、数字でも、グラフィック記号でも使えます。
また、2つの文字変数を継ぐ事も出来ます。

```
10  A$="I␣"
20  B$="AM␣"          ← 1文字分あける。
30  C$="A  BOY"
40  PRINT A$+B$+C$
50  END
RUN
I  AM  A  BOY
READY
▦
```

CLS, LIST, NEW

BASICには、色々な命令文があります。プログラムをRUNする場合、画面にプログラムが残っていると邪魔で見にくいものです。
画面を消す命令を、プログラムの先頭に入れておくとRUNをした時に、画面上にある表示を全部消してくれます。

今のプログラムに、新しい文を入れます。

ＣＬＳ

５　ＣＬＳ　　← 画面の表示を消す命令

今まで入力したプログラムの下の方で、横方向に何も表示がない所に、行番号５　ＣＬＳを入力して下さい。

プログラムの入力は、行番号１から６５５３５まで使えます。

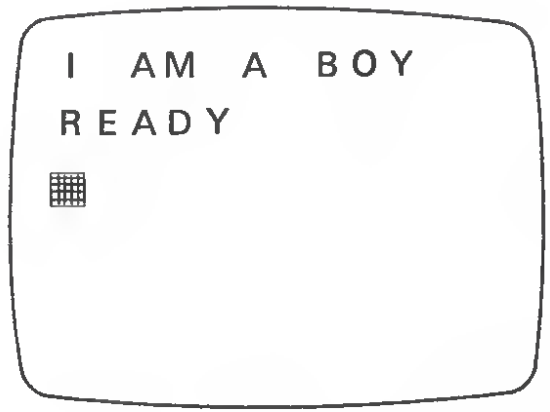
しかし、1, 2, 3, と行番号を詰めてプログラムを作りますと、すでに入力したプログラムの間に追加が出来ません。ですから、追加が出来る様に、行番号は10番置きにすると追加がしやすくなります。
プログラムを、10番置きに行番号100まで入力している時、行番号25や55など、まだ使っていない行番号を、行番号100の後に入力しても、コンピュータは、内部で並べ替えてくれます。
さき程の、5　CLS がどうなっているか調べてみましょう。

ＬＩＳＴ

HOME/CLR　　ＬＩＳＴ　　ＣＲ　　と入力します。

```
  5  CLS
10  A$="I  "
20  B$="AM  "
30  C$="A  BOY"
40  PRINT  A$+B$+C$
50  END
```

行番号5が先頭に入っていますね。
これで実行してください。



LISTは、入力したプログラムを表示させる命令です。

LIST の使い方は、次の方法があります。

ＬＩＳＴ	プログラムの内容を全部表示
ＬＩＳＴ　行番号	１行のみ表示
ＬＩＳＴ　行番号－行番号	行番号から行番号までを表示
ＬＩＳＴ　行番号－	行番号から後ろのプログラムの内容を表示
ＬＩＳＴ　－行番号	プログラムの先頭から行番号までを表示

LIST で表示されたプログラムは、カーソルを使って、内容を書き替えられます。

```
ＬＩＳＴ　３０
３０　Ｃ＄＝〃Ａ　ＢＯＹ〃
```

カーソルをＢの上まで移動して、Ｍを入力します。続いてＡＮ と入力して [ＣＲ] を押して下さい。プログラムの内容を書き替えたら、必らず [ＣＲ] キーを押して下さい。

[ＣＲ] キーを押し忘れますと、画面の文字が変ってもメモリー内容は変りません。

リスト表示中に [SPACE] キーを押しますと、リスト表示が一時中断されます。プログラムをすぐに直したいときは、[BREAK] キーを押して修正してください。リスト表示の中断時に [SPACE] キーを押すと、表示が再開されます。

NEW

一つのプログラムが終って、次に新しいプログラムを入力する場合、前のプログラムがメモリー内に残っていると、プログラムが正常に働かない場合があります。
前に入力したプログラムを消すには、ＣＬＳ文ではメモリー内部から消えません。

プログラムを消すには、

ＮＥＷ と入力して [CR] を押して下さい。

LIST を出してみましょう。

```
ＬＩＳＴ  [CR]
ＲＥＡＤＹ
▦
```

何も表示されませんね。プログラムが、全部メモリーから消えてしまいました。

今までの説明で、少しコンピュータのプログラムが判って来ましたか。
プログラムを入力している時、馴れない内はミスタイプや、文を間違える場合があります。そのままRUN させますと、ミスのある行番号の所で、実行が止まります。このようなミスを、BUG (バグ) と呼びます。バグとは、虫の事で、虫喰いプログラムと言います。
短いプログラムの場合は、バグを見つけるのは簡単ですが、長いプログラムでは見つけるのが大変です。入力する場合は、落ちついてしましょう。

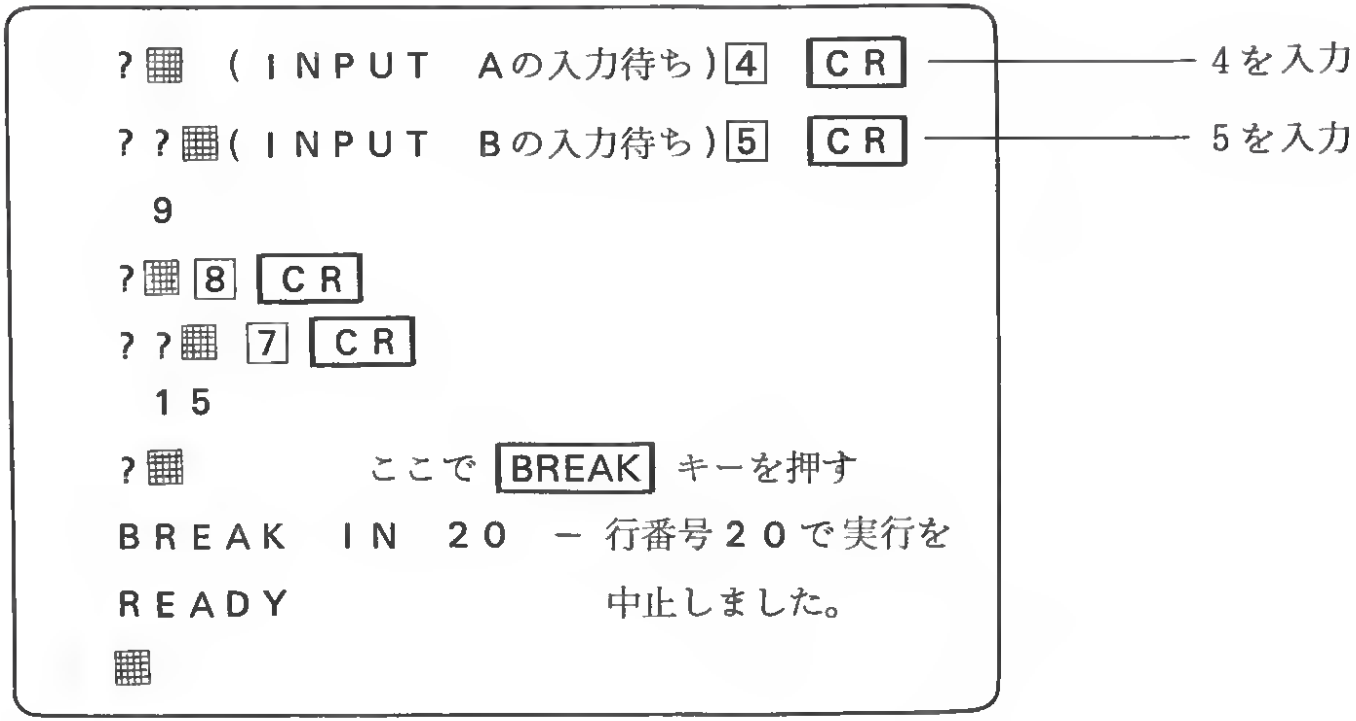
INPUT. GOTO

計算のプログラムを作ってみましょう。

一番始めに作ったプログラムは、変数の値をプログラムとして置いたので、計算したい数をその都度、修正しなければなりませんでした。これでは、電卓より不便です。

では、連続して使える計算プログラムを作ってみましょう。

```
10  CLS
20  INPUT  A
30  INPUT  B
40  C=A+B
50  PRINT  C
60  GOTO  20  ←――― 飛び先の行番号
RUN  [CR]
```


? ▦ （INPUT Aの入力待ち）4 CR
? ? ▦（INPUT Bの入力待ち）5 CR
9
? ▦ 8 CR
? ? ▦ 7 CR
15
? ▦
ここで BREAK キーを押す
BREAK IN 20
行番号20で実行を
READY
中止しました。
▦
4を入力
5を入力

GOTO

プログラムの流れが、GOTO文に出合うと指定された行番号に、無条件で飛びます。
このプログラムは、行番号60から行番号20に戻り、再び、INPUT A から始まります。この様なプログラムは、無限にグルグル廻りますので無限ループと呼ばれ終りがありません。プログラムを止めるには、 BREAK キーで止めるしかありません。
プログラムの流れが、INPUT文に出合うと、キーボードから変数に数値が入力され、 CR キーが押されるまでプログラムの流れが止まっています。
INPUT文は、文字変数も使えます。また、PRINT文と同じに、" " で囲んだ文字も付けられます。

```
NEW
10  CLS
20  INPUT"ナマエハ?␣";A$
30  PRINT  A$
40  END
RUN
ナマエハ?  ハナコ      ←文字を入力
ハナコ
READY
▦
```

END, STOP

ENDはプログラムの終りを知らせます。
STOPは、プログラムの流れを止めます。

CONT

STOP文や、BREAKキーで中断されたプログラムを続けて実行したいときに使います。

```
10  X=X+1
20  PRINT  X
30  GOTO  10
```

このプログラムをRUNさせ、途中で [BREAK] キーで実行を止めて下さい。

```
Break  in  20
CONT  [CR]
```

これで再びプログラムが止まったところから再開されます。

FOR～TO, NEXT, STEP

一定回数、繰り返して仕事をさせる時に使います。

```
10  CLS
20  FOR  N=0  TO  9
30  PRINT  N
40  NEXT  N
50  END
```

10回くり返します。

FOR～TOは、NEXTと一対で使います。このプログラムは、Nが0から9まで1つずつ増えます。

```
10 CLS
20 FOR N=0 TO 20 STEP 2
30 PRINT N;
40 NEXT N
50 END
RUN
0  2  4  6  8  10  12  14  16  18  20
```

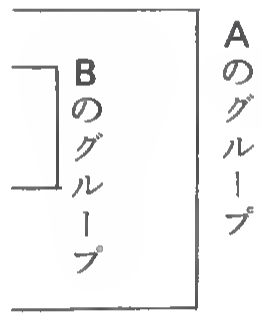
STEP 2 は、きざみ幅を2にしましたので、０～２０を2づつ増しています。
STEPは、－（マイナス）も使えます。行番号20を修正してみましょう。

```
20 FOR N=20 TO 0 STEP-2
RUN
20  18  16  14  12  10  8  6  4  2  0
```

２０から、2つづつ少なくなりましたね。このように一定数、増減させる場合に使います。

FOR～NEXT文は、重ねて使えます。

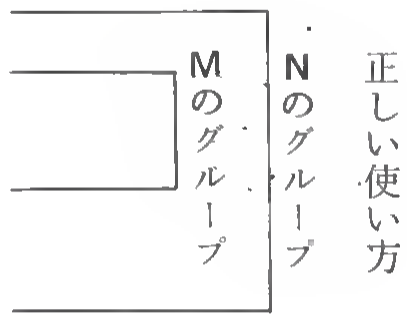
```
10  CLS
20  FOR  A=1  TO  9
30  FOR  B=1  TO  9
40  PRINT  A*B;
50  NEXT  B
60  PRINT
70  NEXT  A
```



FOR～NEXT文を重ねるのを、"入れ子"といいます。4重まで"入れ子"に出来ます。
それ以上重ねると、ネスティング・エラーになります。

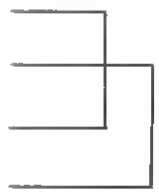
FOR I=1 TO N のように、変数も使えます。

```
10  CLS
20  FOR  N=1  TO  20
30  FOR  M=1  TO  N
40  PRINT"●";
50  NEXT  M
60  PRINT
70  NEXT  N
```



入れ子にする場合の注意

```
FORN=1  TO  20
FORM=1  TO  10 .....
NEXT  N
NEXT  M
```



正しくない使い方

FOR, NEXTのグループを交差させることは出来ません。

IF～THEN. GOSUB

プログラムの流れは、行番号の小さい側から順番に進みますが、ある条件になった場合に、流れを変えるようにしてみましょう。

【合格　不合格の適性プログラム】

```
10  CLS
20  INPUT"トクテン";A
30  IF  A>=65  THEN  GOSUB  100
40  IF  A<65  THEN  GOSUB  200
```

```
50  GOTO  20
100  PRINT"ゴウカク"
110  RETURN
200  PRINT"フゴウカク"
210  RETURN
```

IF～THENは、条件判断をする命令です。上のプログラムは、INPUT文で入力された得点を、IF文で判断して流れを変えております。
入力された得点が、65点以上の場合は、GOSUB文で行番号100に行き、65点に満たない場合は、行番号200に行きます。

```
(もし)              (ならば) (行ってから戻れ) (行番号)
IF  A>=65  THEN  GOSUB  100
```

IF～THENの後には、行番号以外の文も使えます。

```
                (省略出来る)
IF～THEN  GOTO   行番号  (指定行番号にとびます。)
IF～THEN  GOSUB  行番号(指定行番号にとびます。)
IF～THEN  PRINT"×××"  画面に出力します。
IF～THEN  END      (プログラム終了)
IF～THEN  STOP     (プログラム実行中止)
IF～THEN  BEEP     (音を出します)
```

条件判断は表の比較演算子を使って下さい。

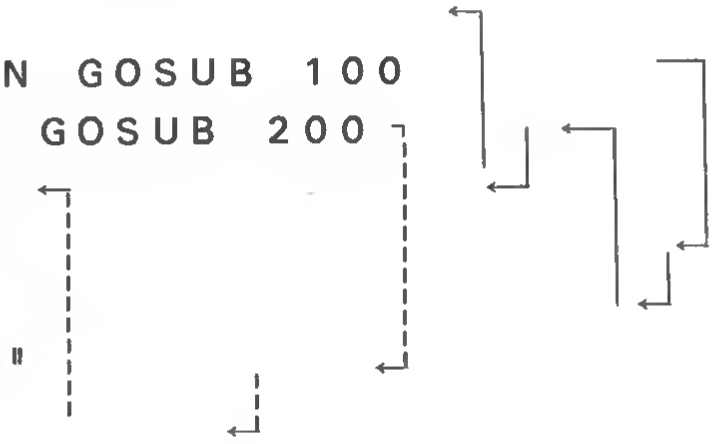
比較演算子表

	記号	内　　容
比較演算	＝	等しい　（True なら－１，False なら０）
	<>	等くない（True なら－１，False なら０）
	＞	大きい　（True なら－１，False なら０）
	＜	小さい　（True なら－１，False なら０）
	＞＝	以　上　（True なら－１，False なら０）
	＜＝	以　下　（True なら－１，False なら０）
論理演算	NOT	論理反転
	AND	論理積
	OR	論理和
	XOR	排他的論理和

GOSUB, RETURN

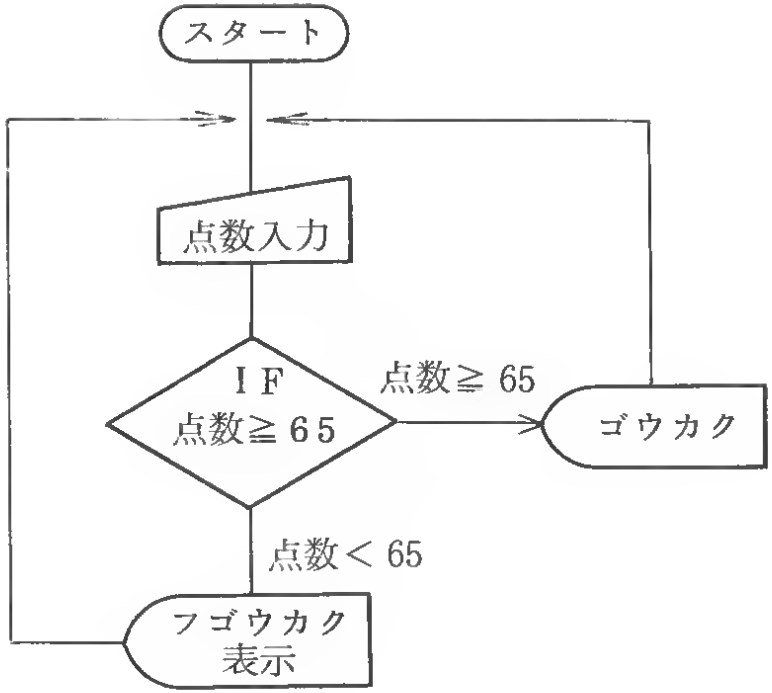
前のプログラムに、GOSUB文がありました。
GOTO 文の場合は、指定された行番号に行くだけでしたが、GOSUB 文は RETURN 文と一諸に使われ、指定された行番号に飛んでから、RETURN 文で GOSUB 文の次の行に戻って来ます。

```
20  INPUT  A
30  IF  A>=65  THEN  GOSUB  100
40  IF  A<65  THEN  GOSUB  200
50  GOTO  20
100  PRINT"ゴウカク"
110  RETURN
200  PRINT"フゴウカク"
210  RETURN
```



GOSUB 文を使う時に、RETURN 文を忘れると結果がおかしくなります。
RETURN 文で戻った後、GOSUB の次の行からプログラムは進行しますので、行番号 50 のように、もう一度、プログラムの流れを変えて下さい。

プログラムが、途中から分岐する場合、流れがよく判るように、フローチャートを作ります。これは、流れ図とも言ばれます。



フローチャートは、上から下に進みます。
条件判断文で、流れが変ります。複雑なプログラムを作る時は、フローチャートを作ると、流れがはっきり判ります。

ON GOTO

条件文に似た使い方で、ON GOTOがあります。

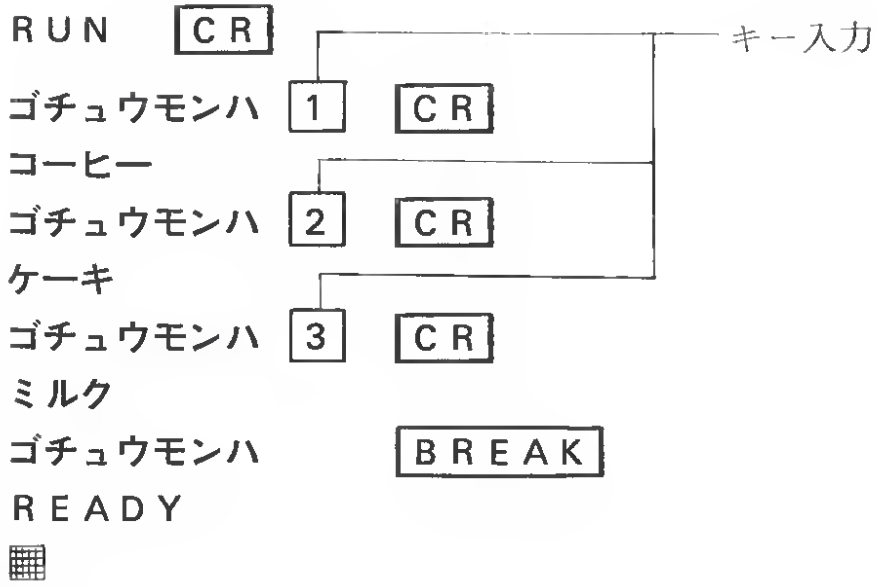
```
ON A GOTO 100,200,300
```

変数Aが、1の時、行番号100 に飛びます。
変数Aが、2の時、行番号200 に飛びます。
GOTO文の後に来る、行番号の数だけ変数の値が使えます。
下のプログラムでは、Aは1～3までの値を入力してください。

```
10 INPUT"ゴチュウモンハ";A
20 ON A GOTO 100,200,300
100 PRINT "コーヒー":GOTO 10
200 PRINT "ケーキ":GOTO 10
300 PRINT "ミルク":GOTO 10
```

(コロン)

1行の中に、2つ以上の命令文を入れる事が出来ます。(マルチステートメント)
独立した文を一行の中に2つ以上入れる場合は、:(コロン)で区切って下さい。

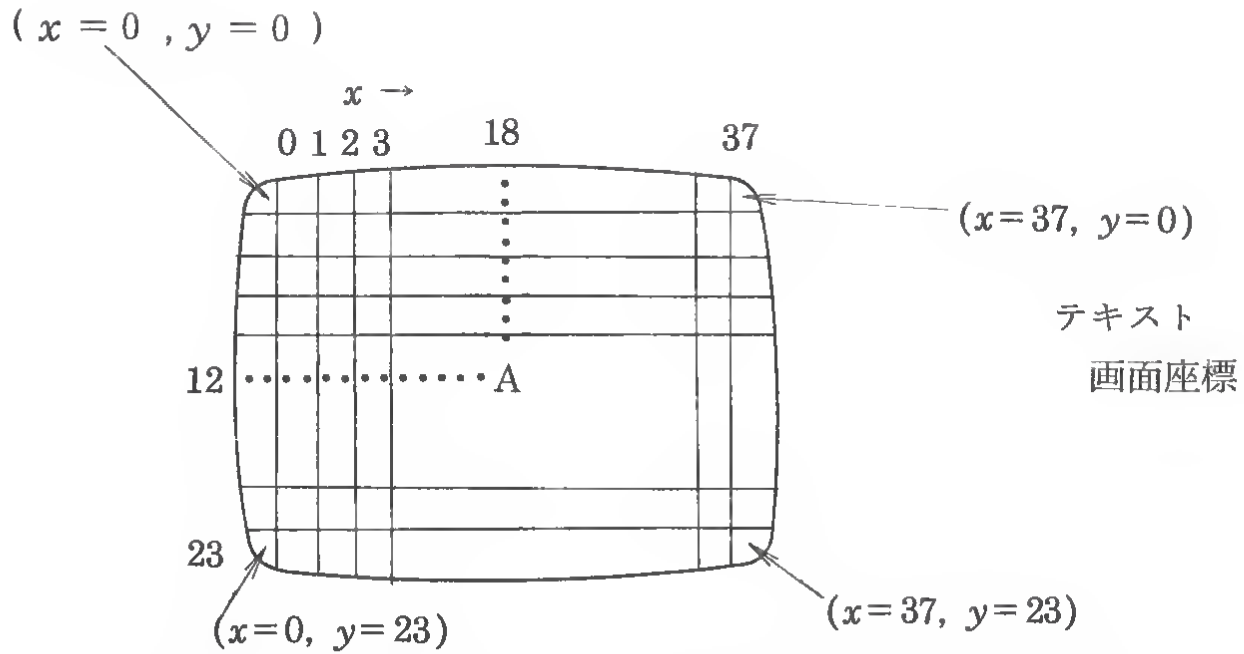


ON GOSUB も、同じ使い方をします。説明の前に新しい命令を使ってみます。

CURSOR

今まで、RUN した時、表示が左側に出るだけでした。これを、画面上のどの位置に出すか、プログラム上で指定した方が見やすくなります。

CURSOR文は、表示位置をきめる命令です。

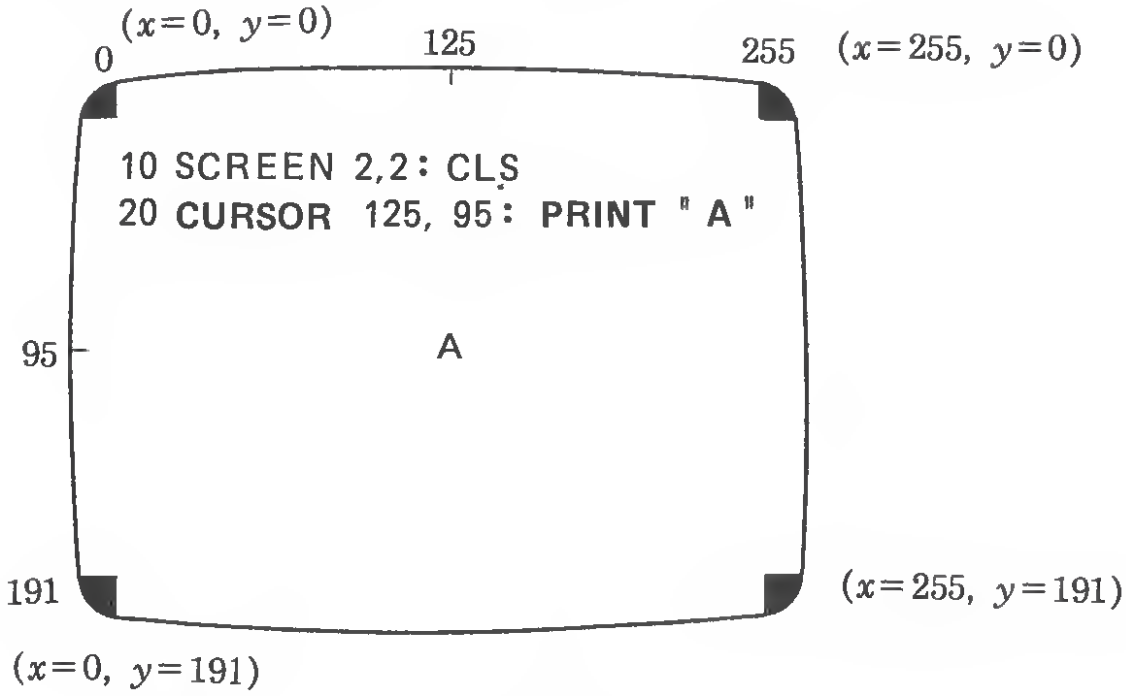


プログラムを書き込む画面（テキスト面）は、38桁 × 24の912のマス目で構成されています。

```
CURSOR 18,12 :PRINT"A"
```

行番号を使わずに、直接入力して下さい。
Aの文字が画面の中央に表示されましたね。

グラフィック画面でCURSOR文を使う場合は、



x 軸方向　0～255　(256ドット)

y 軸方向　0～191　(192ドット)

CURSOR文で座標を指定しますと、表示したい文字の先頭位置がきまります。

ON GOSUB

```
10  CLS
20  CURSOR  10,3:PRINT"メニュー"
30  CURSOR  10,6:PRINT"1...ノミモノ"
40  CURSOR  10,8:PRINT"2...タベモノ"
50  CURSOR  10,10:PRINT"3...デザート"
60  CURSOR  10,13:INPUT"ナニニシマスカ";A
70  ON  A  GOSUB  100,200,300
80  GOTO  60
```

前に表示したものを消す。

```
100  CURSOR  10,16:PRINT"                    "
110  CURSOR  10.16:PRINT"コーヒー...￥300"
120  RETURN
200  CURSOR  10,16:PRINT"                    "
210  CURSOR  10,16:PRINT"ケーキ...  ￥200"
220  RETURN
300  CURSOR  10,16:PRINT"                    "
310  CURSOR  10,16:PRINT"メロン...  ￥250"
320  RETURN
```

GOTO の時のプログラムを、変型したものです。CURSOR 文ばかりですが、表示位置が中央に来ているので、見やすくなります。
GOSUB を使いますので、RETURN は忘れずに入れて下さい。

READ, DATA, RESTORE

プログラムの中に、DATA を入れておき、READ 文で読み出す事が出来ます。

```
10  READ  A,B,C,D
20  PRINT  A+B+C+D
100  DATA  1,2,3,4
RUN
 10
READY
```

DATA の数値を足し算して結果を出しました。
DATA は、文字も扱えます。
プログラムの流れは、READ 文に出あうと、DATA 文がどこにあっても、先に読み取ります。

```
10  READ  A$,B$,C$,D$
20  PRINT  A$+B$+C$+D$
30  DATA  S,E
40  PRINT
50  DATA  G
60  DATA  A
RUN
SEGA

READY
```

1行分のスペースを空ける

文字変数を使った場合、DATAに数字を入れても、文字として扱われて、算数計算には使えません。
DATAの数と、READ文の変数の数は、同数でなければなりません。
DATA数が、多過ぎるとREAD文の変数分だけしか、DATAは出て来ません。
READ文の変数が、DATAより多いとエラーになります。
同じDATAを始めから繰り返して使う場合は、RESTORE文を使います。

RESTORE

```
10  READ  A,B,C,D
20  DATA  1,2,3,4
30  RESTORE
40  READ  E     ← 始めのデータ、1を読む
50  PRINT  A+B+C+D+E
RUN
  11
```

DIM（配列）

一次元配列

前のプログラムでは、DATAに対して、A, B, C, Dと変数を使いました。データ数がふえると、変数を一つづつ設定するのは大変な事です。

この様な場合、配列を使います。

DIM　A（５）　←カッコの中の数値を添字と言います。

この意味は、

A（０），A（１），A（２），A（３），A（４），A（５）

６ケの変数を宣言した事になります。

文字変数も配列宣言が出来ます。

```
10  CLS
20  DIM  A$(5),B(5)
30  FOR  I=0  TO  5
40  READ  A$(I),B(I)
50  PRINT  A$(I),B(I)
60  PRINT  ← 一行、間を空けるため、
70  NEXTI
100  DATA  コーヒー,300,ミルク,150,ケーキ,200
110  DATA  ティー,280,トースト,180,パン,100
```

DIMA $(5)としたのに、READ文でA$(I)になっているのは、FOR I =0 TO 5で使っている、Iです。ですから、Iが0から5に変化しながら、DATAを読み込んでいます。

READ文では、A$とBを続けて読み込みますから、DATAは、文字変数と、数値の変数を交互に並べます。
DATAの並べ方は、実験してみて下さい。

二次元配列

二次元配列は、DIMA(5, 5) のように、カッコの中の添字が2つに分れます。

九九の表

```
10  CLS
20  DIM  A(5,5)
30  FOR  J=1  TO  5
40  FOR  K=1  TO  5
50  A(J,K)=J*K
60  PRINT  A(J,K);
70  NEXT  K
80  PRINT  ← 行を変えるため
90  NEXT  J
RUN

1    2    3    4    5
2    4    6    8   10
3    6    9   12   15
4    8   12   16   20
5   10   15   20   25
```

三次元配列

ＤＩＭ　Ａ（５，５，５）

配列宣言は、３次元までです。

ＤＩＭ文で配列宣言しない場合は、自動的に添字の最大値は 10 です。

ERASE

プログラムの途中で、配列宣言を無効にしたい時に使います。

１００　ＥＲＡＳＥ　としますとプログラム内部の配列は、全部無効になります。

１００　ＥＲＡＳＥ　Ａ，Ｂ＄　のように、配列名を入れますと、その配列を無効にします。

ＤＥＬＥＴＥ（デリート）

プログラムを修正する場合、不要な行を消す時に使います。

ＤＥＬＥＴＥ　１８０－２２０　[CR]　，（コンマ）でも良い、行番号１８０－２２０までを取り消します。

ＤＥＬＥＴＥ　－２５０　先頭から、行番号２５０までのプログラムを消します。

ＤＥＬＥＴＥ　６００－　行番号600から後を全部消します。

ＤＥＬＥＴＥ　１００　行番号100だけ消す。

今まで、BASICの文の一部を使って来ました。
プログラムを作る場合、必らず行番号を入れます。行番号を自動的に作り出す文があります。

AUTO（オート）

AUTO　CR　と、行番号なしで入力して下さい。

```
10    CR
20
```

行番号が、10行きざみで自動的に発生して来ます。

```
AUTO  100    CR
100    CR
110
```

行番号が、100番から、10きざみで出て来ましたね。

```
AUTO,10,20    CR
```

（始めの行番号、きざみ幅）

```
10    CR
30    CR
50
```

CR キーを押すたびに、20きざみで行番号が出て来ます。

RENUM（リナンバー）

プログラムを入力していて、行番号を追加している内に、詰め過ぎた場合に、行番号を付け直したい場合に使います。

RENUM [CR]

プログラムの先頭から、10，20，30 と行番号を付け直します。

RENUM 100 [CR]

行番号100から、10きざみに直します。

RENUM 300，200

新番号 └旧番号

行番号200からのプログラムを、行番号300から10きざみにします。

RENUM 300，200，50

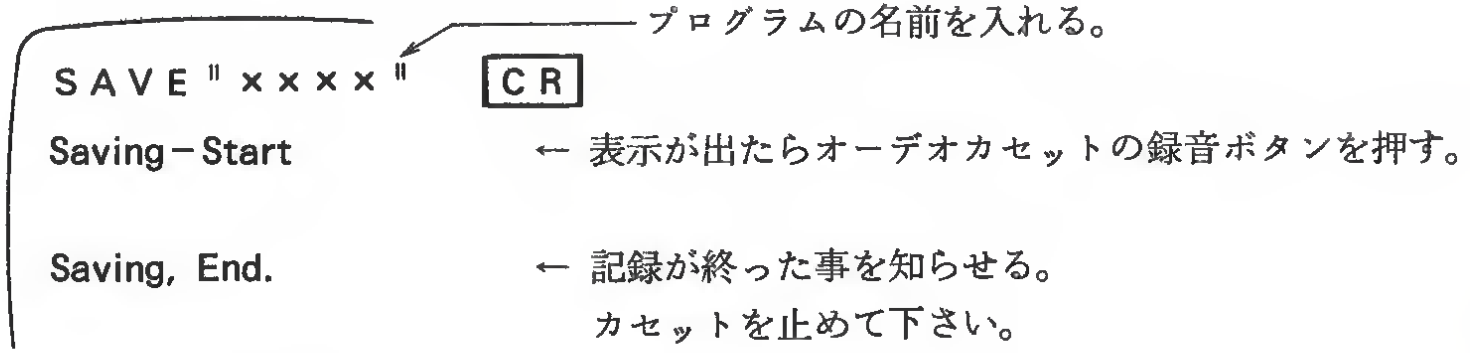
└きざみ幅指定

行番号300から50きざみにします。

SAVE，LOAD，VERIFY

SAVE

作成したプログラム、データ等を作成したプログラムを、カセットテープに記録して置く事が出来ます。



プログラムの内容が判る名前を付けて下さい。（ファイルネーム）

名前は、16字以内で付けて下さい。
カセットデッキは、お手持ちのものをお使い下さい。
カセットデッキに、カウンターが付いていましたら、カウンターの数字をカセットテープのラベルに記入しておいて下さい。

VERIFY

ＳＡＶＥが正確に行われたかをチェックします。

テープを巻き戻し、

ＶＥＲＩＦＹ　[ＣＲ] と入力して、プレイボタンを押して下さい。
正しく、ＳＡＶＥされていれば、ＶＥＲＩＦＹ　ＯＫ　と表示されます。
ＯＫが出ない場合は、始めから、ＳＡＶＥ仕直して下さい。

LOAD

カセットテープに入っているプログラムデータをコンピュータに移します。

ＬＯＡＤ＂×××＂[ＣＲ]　←プログラム名

Loading Start　← プレイボタンを押す。
Found ＂×××＂
Loading End　← カセットを止めて下さい。

お手持ちの、カセットデッキを使う場合、音のレベルにより、カセットテープに書き込みや、読み取りが出来ない場合があります。
これは、コンピュータの故障ではなく、カセットデッキの性能による場合があります。

音量や音質のレベルを変えて試して下さい。それでも書き込みが出来ない場合、コンピュータ対応のカセットデッキを使用して下さい。

カセットテープレコーダとの接続は、市販品のミニプラグを使って下さい。

SC-3000 側		カセットレコーダ側
IN ←	LOAD. VERIFY	→ イヤホーン (EAR)
OUT ←	SAVE	→ マイク (MIC)

REM

プログラムを作成する時にプログラムの内容や、プログラム中のサブルーチンの内容が判るように、注釈文を入れておくと後でプログラムリストを見たとき便利です。

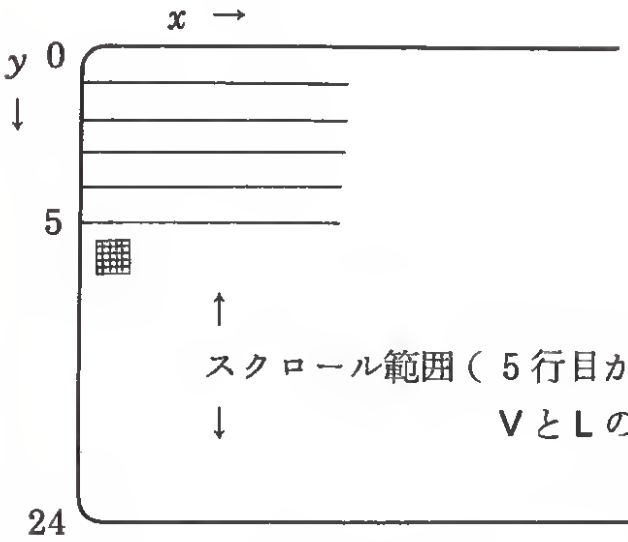
```
10 REM ××× ケイサン ×××
20 CLS
30 PRINT 2+3
~
```

REM文は、プログラム中では無視されて実行はされません。

CONSOLE

テキスト画面のスクロール範囲、クリック音のON, OFF, 英文字の大小文字の切り替えを指定します。

スクロール範囲指定



↑
スクロール範囲（5行目から15行進んだ所までです。）
↓　　VとLの合計が24を越えるとエラーになります。

	V	L	C	S
CONSOLE	5	,15	, 0	, 1

V; スクロールの上限 (0～22)
L; 〃 長さ (2～24)
C; クリック音の有無 0;無し 1=有り
S; 英数の大小 0; シフトなし大文字 1; シフトなし小文字

画面は、y 方向に 0～23行に割られております。CONSOLEの後の数字は、スクロール開始行、スクロール終り行を示します。
例のように設定しますと、5行目から15行目までの間だけカーソルが移動します。
カーソン移動範囲の設定は、0から23までにしてください。24以上を設定しますと、エラーになり

ます。
スクロールの長さは、2から２４（上限の設定数）まで使えます。

CONSOLE ,, 0　3番目の数字は、キーを押した時のクリック音を ON, OFF します。

0；　音が出ない。
1；　音が出る。

CONSOLE ,,, 1　4番目の数字は、英字の大小を決めます。

0；　シフトなしで大文字
1；　シフトなしで小文字

4 つの数字は、省略する場合（　，　）だけを入力して下さい。
CONSOLE文は、RESET キーで解除出来ます。

第４章　　関　　数

関数のグループには、数学的関数と文字関数があります。良く使われる関数を覚えましょう。

ＲＮＤ（ランダム）

変数の値をランダムに発生させます。これの利用方法は、サイコロの目を出したり、ゲームで標的を不規則に動かしたりするのに便利なのでよく使われます。

ＲＮＤ（１）	続けて乱数が発生する。
ＲＮＤ（０）	乱数系列が初期化する。
ＲＮＤ（－１）	乱数系列を入れ変える。

乱数を出してみましょう。

```
10  FOR  N=0  TO  10
20  R=RND(1)
30  PRINT  R
40  NEXT  N
RUN
```

```
0．2380546294
0．7041382496
0．4925371138
```

0から1未満までの間の数字がデタラメに出て来ましたね。このデタラメが乱数なのです。次に出て来る数字が予測出来たのでは、乱数といえません。サイコロの目も振ってみなければ何が出るか、わからないから面白いのです。
でも小数以下の数字では実用的ではありません。これをもっと使いやすくします。

INT（ｎ） インテジャー

ＩＮＴ関数は、小数点のある実数を整数に変えます。

```
？ＩＮＴ（3.14）
 CR
3  ← 小数が消える。
ＲＥＡＤＹ
```

サイコロ

```
１０ ＦＯＲ Ｎ＝０ ＴＯ ２０
２０ Ｓ＝ＩＮＴ（ＲＮＤ（１）＊６）
３０ ＰＲＩＮＴ Ｓ；
```

```
40  NEXT  N
RUN
3  5  0  4  1  5  0  ……
```

小数点がなくなりましたが、０があって６がありませんから直しましょう。

```
20  S＝INT（RND（１）＊６）＋１
```

＋１ ―― ０を出さない為と必要な数までにするため

６ ―→ 出したい数

こんどは、１から６までの数が出ましたね。数値を色々変えて試して下さい。

四拾五入プログラム

```
10  INPUT  A  ← 小数付き数値
20  PRINT  INT（A＋０．５）
30  GOTO  10
```

小数点以下を四拾五入するプログラムです。これは、いろいろな計算に応用が出来ますので覚えましょう。

文字列関数

前におぼえた文字変数についてもう少しくわしく説明しましょう。

文字変数は、文字列変数ともいいます。他のテキストでは、ストリング変数と書いたのもあります。

コンピュータの中では、文字はどのようにあつかわれているか調べてみましょう。

ＡＳＣ（＂ｎ＂）　アスキー関数

ASCII（アスキー）とは、American Standard Code Information Interchange（アメリカンスタンダード　コード　インフォメーション　インターチェンジ）アメリカの基準コードで文字や記号に番号をつけて、コンピュータが情報を処理しやすくされています。

では試してみましょう。

```
？ＡＳＣ（＂Ａ＂）  CR  ダイレクトモードで実行
６５
Ｒｅａｄｙ
```

この６５という数字は、Ａを現わしています。ＡＳＣ文を使うときは（＂Ａ＂）のように、カッコ内の文字を＂＂で囲んで下さい。
Ａの替りに、今度は記号を入れてみて下さい。

？ＡＳＣ（＂！＂）　　今度は３３と出ましたね。

コンピュータの内部には、キーボードにある文字や記号が32番から255番までの数字に対応して入っています。
コンピュータは、人間の様に文字をそのまま判断出来ません。すべて数字として扱います。ですから、Ａ（65）はＢ（66）より先にあると判断して文字を分類することが出来ます。
？ＡＳＣ（＂BA＂）のように、２つ以上入れても始めの一文字だけ調べて数字を出します。

もう一つ試しましょう。

```
１０　ＩＮＰＵＴ　Ａ$
２０　Ｑ＝ＡＳＣ（Ａ$）
３０　ＰＲＩＮＴ　Ｑ
４０　ＧＯＴＯ　１０
```

ＲＵＮしたあと、文字や記号を入力するとそれに対応したコード番号が出て来ます。

ＣＨＲ$

ＡＳＣ文と対になっていて、数値を文字やコントロール機能を与えます。

```
？ＣＨＲ$（６５）［ＣＲ］
Ａ
```

Aは、ASCII コードでは65でしたね。

コンピュータの中に入っている文字をのぞいてみましょう。

```
10  FOR  M=32  TO  255
20  PRINT  CHR$(M);
30  NEXT  M
RUN
```

キーボードに印字されている文字や記号が並んで出て来ましたね。これが、コンピュータの内部にある文字や記号です。
キャラクターセットを見て下さい。コード表と画面上に表示されたコードが同じですね。

LEFT$, RIGHT$, MID$

長い文字列の中から、文字の一部を取り出す関数です。

```
10  A$="コーヒー  ココア  ミルク"
20  M$=LEFT$(A$,4)
30  PRINT  M$
RUN
コーヒー
```

←(左から4番目までの文字)

Ａ＄の中にある文字列（スペースも数えます）の４番目までを取り出して、Ｍ＄の中に入れ画面に表示させます。

```
１０　Ａ＄＝〃コーヒー　ココア　ミルク〃
２０　Ｍ＄＝ＲＩＧＨＴ＄（Ａ＄，３）
３０　ＰＲＩＮＴ　Ｍ＄
ＲＵＮ
ミルク
```

右から３番目までの文字

こんどは、右から３番目を起点として終りまでの文字を取り出します。

```
１０　Ａ＄＝〃コーヒー　ココア　ミルク〃
２０　Ｍ＄＝ＭＩＤ＄（Ａ＄，６，３）
３０　ＰＲＩＮＴ　Ｍ＄
ＲＵＮ
ココア
```

６：起点　３：取り出す文字数

文字列の左から６番の文字を起点として３つの文字を取り出します。

ＬＥＮ（レングス）

ＬＥＮ（Ａ＄）は、Ａ＄の文字数を教えてくれます。この場合も、〃〃で囲まれた文字全部で、

スペースも含まれます。もし" "の間が全部スペースでも文字としてあつかいます。

```
10  A$"SEGA  PERSONAL  COMPUTER"
20  PRINT  LEN(A$)
RUN
22
```

文字の数をスペースを含んで教えてくれました。

こんな使い方もあります。

```
10  A$="**********"
20  FOR  I=1  TO  LEN(A$)
30  PRINT  LEFT$(A$,I)
40  NEXT  I
RUN
*
**
***
****
~
**********
```

STR$, VAL

数値を文字変数に変換したり、文字変数として使われている数字の文字列を数値に変換します。

ＳＴＲ＄

```
１０　Ａ＝１：Ｂ＝３
２０　Ｄ＄＝ＳＴＲ＄（Ａ）＋ＳＴＲ＄（Ｂ）
３０　Ｄ＝Ａ＋Ｂ
４０　ＰＲＩＮＴ　Ｄ＄，Ｄ
ＲＵＮ
１　３　　　４　← 行番号３０の結果
└─ 行番号２０の結果
```

ＳＴＲ＄（Ａ）としますと、数字が文字に変ります。文字の足し算は、文字が並ぶだけで計算としての答は出て来ません。

ＶＡＬ

ＶＡＬ関数は、ＳＴＲ＄とまったく正反対の働きを持ち、文字列の数字を数値に直します。

```
１０　Ａ＄＝"１２３４５"
２０　Ｂ＄＝"１１１１１"
```

```
30  C$=A$+B$        文字列のたし算
40  C=VAL(A$)+VAL(B$)  ← 数値のたし算
50  PRINT  C$
60  PRINT  C
RUN
1234511111  ← 文字列
  23456     ← 数値
```

TIME$

コンピュータの内部には、時計機能があります。水晶発振の正確なクオーツ．デジタル時計です。
コンピュータの電源を ON しますとその時から1秒きざみで働き始めます。

電源を入れた時　00：00：00　です。

一定時間が過ぎると、その時間だけ表示します。

```
PRINT  TIME$   CR
  00：12：32  ← 電源を入れてからの時間
```

時計として使う場合は、

```
10  TIME$ ="08:15:00"  ←現在の時刻（"時:分:秒"）
20  CURSOR  15,15:PRINT  TIME$
30  GOTO  20
```

一度時刻を入力しますと、Reset ボタンを押すか、電源を切るまで残っています。
これでデジタル時計になりました。

SPC（スペース），TAB（タビュレーション）

PRINT文で使います。

SPC関数は、文字と文字の間のスペースを指定します。

```
10  PRINT"ABC";SPC(10);"XYZ"
RUN
ABC ␣␣␣␣␣␣␣␣␣␣ XYZ
                  10ヶの空白（スペース）
```

SPCで指定した範囲に文字があると消されてしまいます。

TAB関数は、画面の端から何文字目に出すか指定します。

```
10  PRINT  TAB(5);"ABC"
RUN
```

␣␣␣␣␣ＡＢＣ

５文字分のスペース

ＴＡＢ関数の場合は、指定した場所までの間に文字があっても消しません。
この関数は、ＰＲＩＮＴ文の時に使いますのでよく覚えましょう。

ＩＮＫＥＹ＄

文字や数字のどのキーが押されたかを見る文です。
ゲームのようにキーボードで何かの絵（キャラクター）を動かすのに便利です。

```
１０　Ｘ＄＝ＩＮＫＥＹ＄
２０　ＩＦ　Ｘ＄＝"　"　ＴＨＥＮ　１０
３０　ＰＲＩＮＴ　Ｘ＄；
４０　ＧＯＴＯ　１０
```

行番号２０でキーが押されているかを見ています。何も入力されていないときはＸ＄の値はヌル（何もしない事）ストリングといって、何も表示もせず行番号１０と２０の間をぐるぐる廻っています。（無限ループ）
もし何かのキーが押されますと、そのキーの値がＸ＄の中に入り行番号３０で表示されます。この無限ループから抜けるには、

```
25  IF  X$="Z"  THEN  100
100  PRINT"END":END
```

と追加して下さい。
これでZキーを押すとプログラムの実行が終ります。

例 ←,→キーで、動きます。

```
10 DIM  D (29)
20 CLS
30 X=18: Y=20
40 D (29) = −1 : D (28) = 1 : D (0) = 0
50 K$ = INKEY$
60 IF  K$ = " " THEN  K = 0 : GOTO  90
70 K = ASC (K$)
80 IF  K >29  THEN  K = 0
90 X = X + D (K)
100 IF  X<0  THEN  X = 0
110 IF  X >33  THEN  X = 33
120 CURSOR  X , Y : PRINT "  I + I  "
130 GOTO  50
```

ＦＲＥ

プログラムを入力しますと、残りのメモリーが減っていきます。あと、どれぐらいメモリーが残っているか知りたいときに、ＦＲＥ関数で調べます。

例

```
ＰＲＩＮＴ　ＦＲＥ
　８３００
```

あと８３００ＢＹＴＥ（バイト）のプログラムを入れることができます。

プリンター制御命令

ＬＬＩＳＴ

プリンターにプログラムリストを書きます。

○命令文の使い方は、ＬＩＳＴと同じです。

ＬＬＩＳＴ		プログラム全部を書きます。
ＬＬＩＳＴ	行番号	指定した行番号を書きます。
ＬＬＩＳＴ	行番号 －	行番号
ＬＬＩＳＴ	－ 行番号	
ＬＬＩＳＴ	行番号－	

LPRINT

プリンターにPRINT文の内容を書きます。

○命令文の使い方は、PRINT文と同じです。

```
LPRINT  A     ← Aの内容をプリンターに書きます。

10  INPUT  A,B
20  C=A+B
30  LPRINT  C
40  GOTO  10
RUN
```

プリンターにCの値が書き出されます。

HCOPY

TV画面に表示されている、文字や記号がプリンターに書き出されます。
しかし、プリンターに書けるのは、数字、英字の大小文字、カナ、ASCIIコードの記号のみです。グラフィックモードの記号は書けません。

第５章　　グラフィック

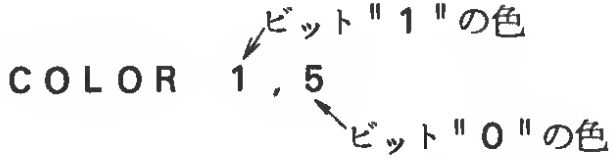
グラフィックを使いましょう。

いよいよ、カラーを使って図形を描いてみましょう。色々な図形や、色を塗ったりするのは案外簡単です。エラーを恐れずに試してみましょう。

COLOR

文字や、キャラクター、下地の色を指定します。

例



テキスト画面（プログラムを入力する画面）

ビット＂１＂の色は文字などの色を指定します。
例では、文字が黒、地は青になります。

グラフィック画面（グラフィック文で表示する画面）

ビット＂１＂の色は、

ＰＲＩＮＴ文による文字の色
ＬＩＮＥ，ＰＳＥＴ 文による線や点の色。
ＬＩＮＥ，
ＰＡＩＮＴ文による塗りつぶしの色。

ビット〝０〟の色
下地の色
ＢＬＩＮＥ，ＰＲＥＳＥＴ，によりビット〝１〟が消された時の色。

例

```
10 CLS                          40 FOR I=0 TO 300
20 FOR C=0 TO 15                50 NEXT I,C
30 COLOR 1,C
```

画面の色が次々と変ります。

色番号	色	色番号	色	色番号	色
0	透　明	6	こい赤	12	こい緑
1	黒	7	水色(シアン)	13	マゼンダ
2	緑	8	赤	14	灰
3	うすい緑	9	うすい赤	15	白
4	こい青	10	こい黄		
5	うすい青	11	うすい黄		

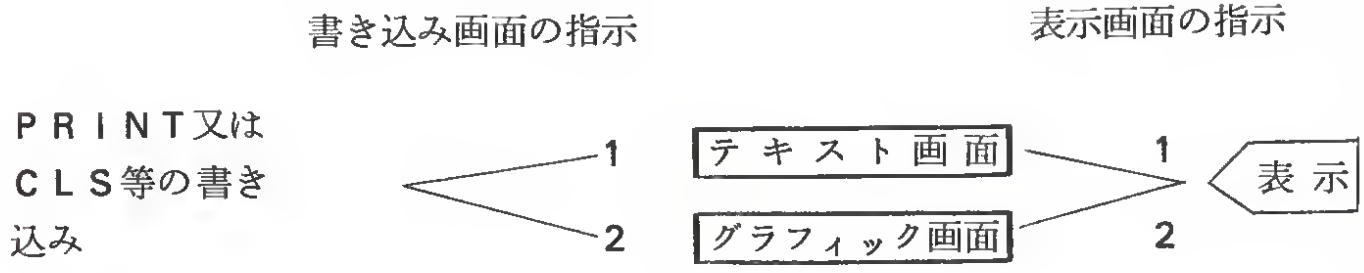
SCREEN

書き込み画面、表示画面を選びます。

テキスト画面しか使わないのでしたら、ＳＣＲＥＥＮ指定はいりません。
グラフィック面に文字や絵を表示するときに使います。

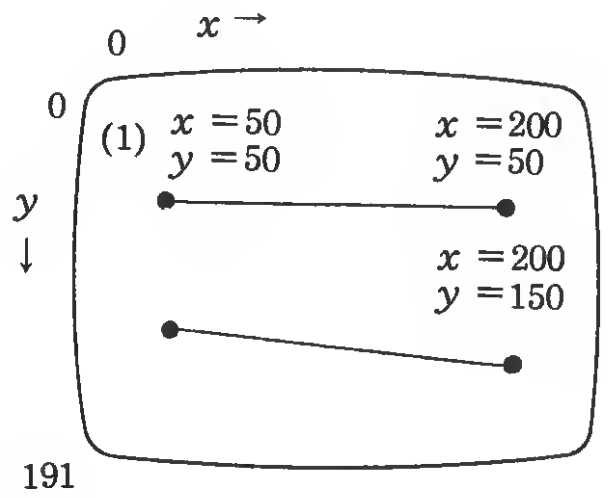
ＳＣＲＥＥＮ　書き込み画面、表示画面
　　＊

＊ ┌ １：　テキスト画面
　 └ ２：　グラフィック画面



ＬＩＮＥ

線を引きます。(SCREEN 2,2を入力してから使います)



グラフィック画面は、x方向は0から255（256ドット）y方向は0から191（192ドット）の座標を持っています。

ＬＩＮＥ文は、２点間の座標を指定して線を引きます。

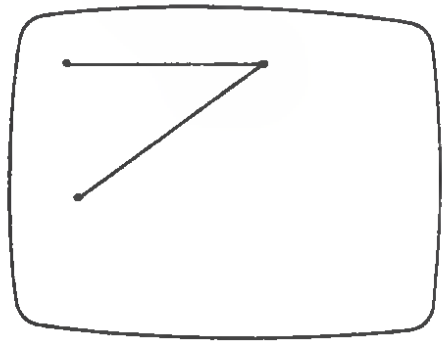
画面の例

（１）ＬＩＮＥ（５０，５０）－（２００，５０），１　　黒色指定

（２）ＬＩＮＥ（５０，１００）－（２００，１５０），８　赤色指定

ＬＩＮＥ文を連続して使う場合、書き始めの座標を省略しますと以前に描いた座標から線を引きます。

```
１０　ＳＣＲＥＥＮ２，２：ＣＬＳ
２０　ＬＩＮＥ（５０，５０）－（１５０，５０），１
３０　ＬＩＮＥ－（５０，１５０），８
```



線を描き終ると、すぐにテキスト画面に変ります。ＳＨＩＦＴキーを押して [↘] キーを押して下さい。画面が変りグラフィック面に線が描かれてるのがわかります。

箱を描く（ＢＯＸ）

ＬＩＮＥ文で対角線を指定して、色の後にＢを付けて下さい。

```
１０　ＳＣＲＥＥＮ　２，２：ＣＬＳ
２０　ＦＯＲ　Ｃ＝０　ＴＯ　１５
３０　ＬＩＮＥ（８０，５０）－（１６０，１００），Ｃ，Ｂ
４０　ＦＯＲ　Ｉ＝０　ＴＯ　３００：ＮＥＸＴ　Ｉ
```

```
50  NEXT  C
60  GOTO  60  （無限ループにしました。BREAKキーで止まります）
```

箱の中を塗りましょう。行番行30のBの後にFを入れて下さい。Cの色指定できれいに塗り変えられていきます。

BLINE

LINE文で描いた線や、箱を消します。使い方は、LINE文と同じですが色指定は無視されますのでいりません。

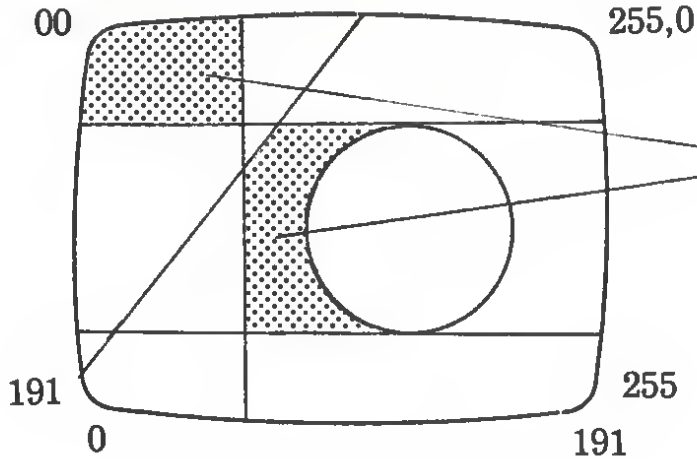
```
10  SCREEN2,2:CLS
20  LINE(50,50)-(200,50),1
30  FOR  I=0  TO  300:NEXT  I  ← 時間をとる。
40  BLINE(50,50)-(200,50)
50  GOTO  50
```

箱を消す時も同じですが、描いた箱より小さい箱をBLINEで描きますと、その部分だけ色が消えます。

BLINE文は、一度描いた絵を全部消したり、一部分だけ消したりするのに使います。

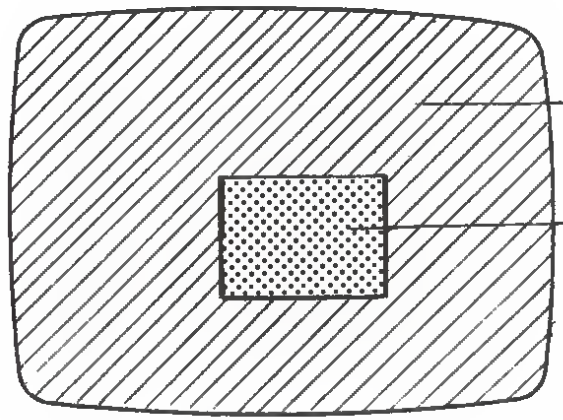
PAINT

画面に LINE 文や CIRCLE 文で区切られた部分を塗ります。



PAINT（ x , y ）, 色

塗り始めの座標線で囲まれている部分を塗ります。



まわりを全部塗ります。

LINE , BF で描いた箱

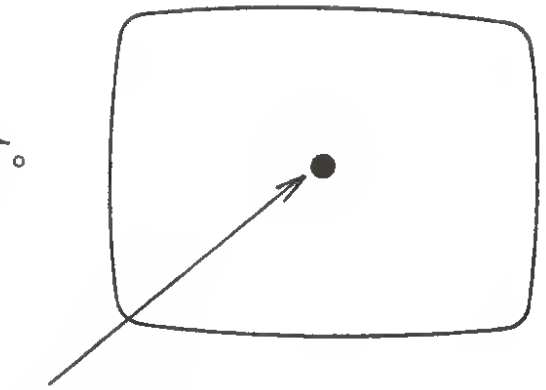
線で区切る場合、すき間を作らないようにして下さい。

PSET

画面上の指定された位置に点を打ちます。

座標　　色

PSET（x ，y ），1



座標を連続的に変化させると直線や曲線が描けます。

```
10  SCREEN  2，2：CLS
20  X＝0：Y＝95：E＝1
30  PSET（X，Y），8
40  X＝X＋1：Y＝Y＋E
50  IF  Y＝120  THEN  E＝－1
55  IF  Y＝85  THEN  E＝1
60  IF  X＝250  THEN  END
70  GOTO  30
```

PSETは、点をちりばめたり関数を使ったグラフを作ったりします。

PRESET

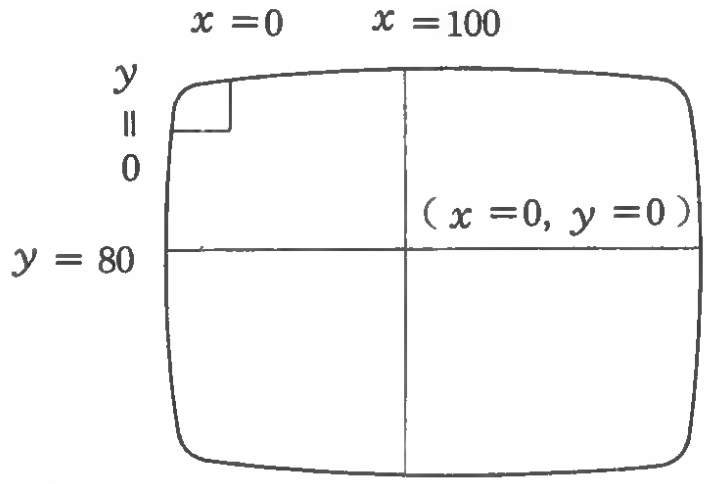
PSETと反対に点で消します。使い方はPSETと同じで点を描く替りに消す役目です。使い方は同じです。

座標

PRESET（x，y）

POSITION

画面の座標は左上がx＝0，y＝0です。
POSITION文で座標を指定すると、その位置が中心点、x＝0，y＝0になります。



x軸　y軸

POSITION（１００，８０）０，０

１　１—yの軸方向

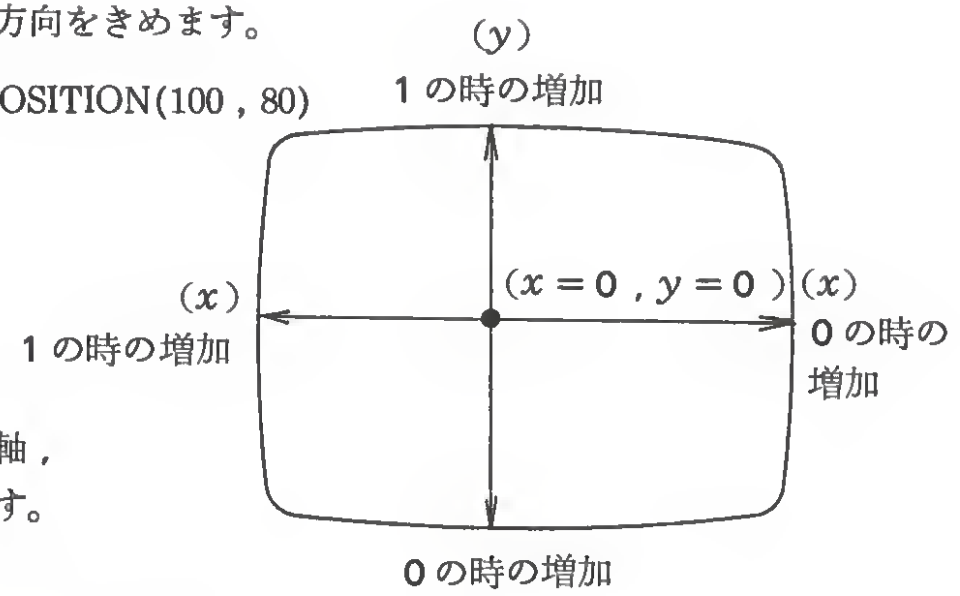
└—xの軸方向

座標の後の数値は、x 軸 y 軸の増加方向をきめます。

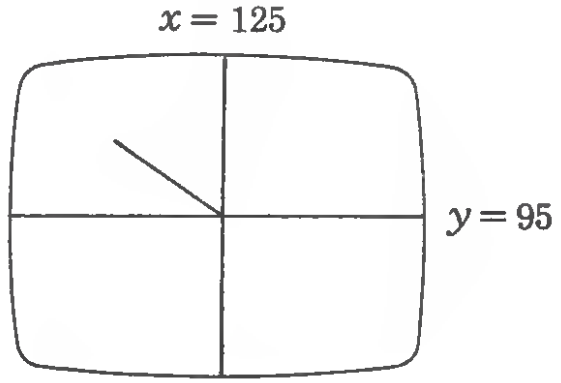
0，指定　x は右に増加
y は下方に増加します。

1，指定　x は左に増加
y は上に増加します。

x，y の指定を組み合せると x 軸，
y 軸の増加分を変化させられます。




```
10  SCREEN  2，2：CLS
20  POSITION（125，95），1，1
30  FOR  N＝O  TO  50
40  PSET（X，Y），1
50  X＝X＋1：Y＝Y＋1
60  NEXT  N
```



POSITION文は、PSETと一諸に使うと関数グラフなどが描けます。

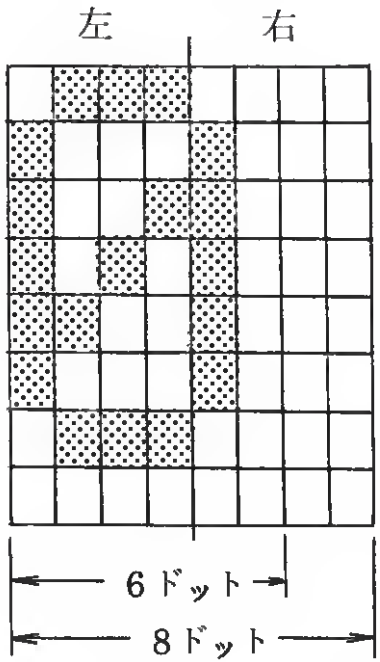
```
10 SCREEN 2,2:CLS
20 POSITION(100,50),0,0
30 FOR N=-10 TO 1 STEP.1
40 X=N*20+120:Y=SIN(N)*50+45
50 PSET (X,Y),1
60 NEXT N
```

PATTERN

PATTERN 文を使って文字や絵（キャラクター）を作る事が出来ます。
テキストモードの文字を書き変えてみましょう。

PATTERN　　C＃　文字コード（32 ～ 255 又は、&H20～&HFF），"文字列式"

（文字列式は16進数）



左	右	左	右
0111	0000	7	0
1000	1000	8	8
1001	1000	9	8
1010	1000	A	8
1100	1000	C	8
0111	0000	7	0
0000	0000	0	0

黒いマス＝1，白いマス＝0

図の数字をスペースキーに出すようにしてみましょう。

```
20 PATTERNC#&H20,"708898A8C87000"
RUN
```

これでスペースキーを押してみてください。"0"が出ましたね。スペースに"0"が書き込まれたからです。
このように、文字を作ることが簡単に出来ます。グラフィック画面に出す絵も同じように書きます。

もう一度、使い方を整理します。

◎ C# …… テキストモード指定

◎ 文字コード　16進数の場合、&H20から&HFFまでの数値で入力します。
10進数の場合、32から255までの数値で入力します。

◎ 文字列式　8×8のマス目ドットを黒く塗りつぶして文字や図形を描きます。
黒い点は1，白い点は0として各列毎に1と0を割当てます。

は01110000になります。これを中心から左右半分に分け、16進数に変換します。
0111は7，0000は0で、"70"となります。

このような方法でキャラクターを作ってみてください。

パターンの描き方

まず、グラフ用紙を８×８のマス目に仕切り、ドット（マス目一つ分）を塗りつぶしてパターンを作ります。
塗った所を、１，空白を ０ とします。マス目の横に０や１の数字を並べて下さい。
並んだ８個の数字を真中から二つに分けて四桁づつにします。四桁の数字は２進数をあらわします。これを今度は１６進数に直し２桁の１６進数に直します。
１０進数、２進数、１６進数の対照表を参考にして下さい。
１６進に直された８組の数字が文字列式として＂ ＂の中に代入されます。
これで文字は、８×６で表現されます。

10進数	2進数	16進数
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0 (けた上り)	2
3	0 0 1 1	3
4	0 1 0 0	4
5	0 1 0 1	5
6	0 1 1 0	6
7	0 1 1 1	7
8	1 0 0 0	8
9	1 0 0 1	9
10 (けた上り)	1 0 1 0	A
11	1 0 1 1	B
12	1 1 0 0	C
13	1 1 0 1	D
14	1 1 1 0	E
15	1 1 1 1	F
16	1 0 0 0 0	10 (けた上り)

第6章　関　数　そ　の　2

コンピュータは、計算が得意です。計算機能を高めるために、三角関数を始め種々な関数が組み込まれております。

ABS（X）

機　能　　式Xの絶対値を与えます。
形　式　　ABS（X）
使い方　　PRINT　ABS（－5）[CR]　5
　　PRINT　ABS（3×（－6））[CR] 18

RAD

機　能　　角度からラジアンに変換します。
形　式　　RAD（X）
使い方　　0°，15°，30°，45°，60° をラジアンに変換する。

```
10 FORI=0 TO 60 STEP 15
20 X=RAD(I)
30 PRINT"RAD(";I;"°)=";X
```

```
40 NEXT I
RUN
RAD(0°)=0
RAD(15°)=.261799387 8
RAD(30°)=.523598775 6
```

DEG

機　能　　ラジアンから度に変換します。
形　式　　DEG(X)　　Xはラジアンです。
使い方　　PRINT DEG(0.26) CR (14.896902673)

PI

機　能　　円周率を与えます。
形　式　　PRINT PI CR (3.1415926536)
使い方

```
10 INPUT"ハンケイ";A
20 S=A^2*PI
30 PRINT"エン ノ メンセキ";S
RUN
```

```
? 5
 78.5398163 33
```

ＳＩＮ（サイン）

機　能　　三角関数、正弦の値を与えます。
型　式　　ＳＩＮ（Ｘ）　　引数（Ｘ）はラジアン
使い方

```
10 FOR TH=0 TO 90 STEP 30
20 S=SIN(RAD(TH))
30 PRINT TH;TAB(10);S
40 NEXT TH
RUN
0          0
30         .5
60         .866025403 79
90         1
```

COS（コサイン）

機　能　　三角関数、余弦の値を与えます。
形　式　　COS（X）　　引数（X）ラジアン
使い方

```
10 FOR X=0 TO 90 STEP 30
20 A=COS(RAD(X))
30 PRINT X;TAB(10);A
RUN
 0          1
 30         .866025403 79
 60         .500000000 01
 90         0
```

TAN（タンジェント）

機　能　　三角関数、正弦の値を与えます。
形　式　　TAN（X）　　引数（X）はラジアン
使い方

```
10 INPUT"ドスウ";A
```

```
20  X=TAN(RAD(A))
30  PRINT"TAN(";A;"°)=";X
RUN
ドスウ  30
TAN(30°)=    .57735026919
```

ASN（アークサイン）　　逆正弦

機　能　　SINθのθの値（度数）を求めます。
形　式　　ASN(X)　　Xは-1～1
使い方

```
10  X=ASN(.5)
20  Y=DEG(X)
30  PRINT  Y
RUN
 30
```

ACS（アークコサイン）　　逆余弦

機　能　　COSθのθの値（度数）を求めます。

形　式　　ＡＣＳ（Ｘ）　　Ｘは－１～１

使い方

```
１０　Ｘ＝ＡＣＳ（－１）
２０　Ｙ＝ＤＥＧ（Ｘ）
３０　ＰＲＩＮＴ　Ｙ
ＲＵＮ
　１８０
```

ＡＴＮ（アークタンジェント）　　逆正接

機　能　　逆正接の値を求めます。

形　式　　ＡＴＮ（Ｘ）

使い方　　求められる値は$-\frac{\pi}{2}$から$\frac{\pi}{2}$の間の値です。

```
１０　Ｘ＝ＡＴＮ（Ａ）
２０　ＰＲＩＮＴ　Ｘ
ＲＵＮ
　．７８５３９８１６３４
```

ＳＧＮ（サイン）

機　能　　数値の符号を求める関数です。
Ｘの値が負の時は－１
０の時は　０
正の時は　１

形　式　　ＳＧＮ（Ｘ）

使い方

```
１０　ＦＯＲ　Ｉ＝－２　ＴＯ　２
２０　Ｎ＝ＳＧＮ（Ｉ）
３０　ＰＲＩＮＴ＂ＳＧＮ（＂；Ｉ；＂）＝＂；Ｎ
４０　ＮＥＸＴ　Ｉ
ＲＵＮ
 ＳＧＮ（－２）＝－１
 ＳＧＮ（－１）＝－１
 ＳＧＮ（０）＝０
 ＳＧＮ（１）＝１
 ＳＧＮ（２）＝１
```

ＬＯＧ

機　能　　eを底とする数値の自然対数を求めます。
形　式　　ＬＯＧ（Ｘ）
使い方

```
１０　ＦＯＲ　Ｊ＝１　ＴＯ　３
２０　Ｘ＝ＬＯＧ（Ｊ）　　← 引数Ｊは正の値です。
３０　ＰＲＩＮＴ＂ＬＯＧ（＂；Ｊ；＂）＝＂；Ｘ
４０　ＮＥＸＴ　Ｊ
ＲＵＮ
ＬＯＧ（１）＝２．６７４６８５３２Ｅ－１１
ＬＯＧ（２）＝　．６９３１４７１８０５７
ＬＯＧ（３）＝１．０９８６１２２８８６
```

ＬＴＷ

機　能　　２を底とする常用対数を求めます。
形　式　　ＬＴＷ（Ｘ）
使い方　　ＬＯＧと同じです。

ＬＧＴ

機　能　　１０を底とする数値の常用対数を求めます。
形　式　　ＬＧＴ（Ｘ）
使い方　　１０，１００，１０００の常用対数を求めます。

```
１０　Ｎ＝１
２０　Ｎ＝Ｎ＊１０
３０　Ｘ＝ＬＧＴ（Ｎ）
４０　ＰＲＩＮＴ＂ＬＧＴ（＂；Ｉ；＂）＝＂；Ｘ
５０　ＩＦ　Ｎ＝＜１０００　ＴＨＥＮ　２０
ＲＵＮ
ＬＧＴ（１０）＝１
ＬＧＴ（１００）＝２
ＬＧＴ（１０００）＝３
```

ＥＸＰ

機　能　　自然対数の底eの、べき乗を求めます。
形　式　　ＥＸＰ（Ｘ）
使い方　　e^1．e^2．e^3 をそれぞれ求めてみます。

```
10  FOR  I=1  TO  3
20  X=EXP(I)
30  PRINT"EXP(";I;")";X
40  NEXT  I
RUN
EXP(1)=2.718281828
EXP(2)=7.389056098
EXP(3)=20.08553692
```

SQR

機　能　　数値の平方根を求めます。

形　式　　SQR(X)

使い方　　$\sqrt{2}$, $\sqrt{3}$, を求めてみます。

```
10  INPUT"スウジ";A
20  X=SQR(A)
30  PRINT"ルート";A;"=";X
40  GOTO  10
RUN
```

```
スウジ　2
ルート＝１．４１４２１３５６２４
スウジ　3
ルート＝１．７３２０５０８０７６
```

ＨＥＸ＄

機　能　　１０進数の数値を16進に変換します。
形　式　　ＨＥＸ＄（Ｘ）
使い方　　16進数に変換出来る数値の範囲は－32768～32767の間です。
　　－10，－５，０，５，10，15を16進に変換します。

```
１０　ＦＯＲ　Ｓ＝－１０　ＴＯ　１５　ＳＴＥＰ　５
２０　Ｘ＄＝ＨＥＸ＄（Ｓ）
３０　ＰＲＩＮＴ　Ｓ；＂＝＂；Ｘ＄
４０　ＮＥＸＴ　Ｓ
ＲＵＮ
－１０＝ＦＦＦ６
－５＝ＦＦＦＢ
０＝０　　１０＝Ａ
５＝５　　１５＝Ｆ
```

SOUND

ＳＣ－３０００には、シンセサイザ機能があります。

例

```
SOUND  1,1000,15
       ↑    ↑   ↑
       |    |   └ 音 量
       |    └ 周波数
       └ チャンネル
```

１０００ヘルツの音がでました。

＜チャンネル＞

１つのチャンネルからは、１つの音しか出ません。チャンネルは、０～５までの６個の指定があります。（三重和音まで発生できます。）

０　：音を消します。

例　ＳＯＵＮＤ　０

１～３：１１０Ｈｚ～の音を出します。

４　：ホワイトノイズの選択

5　：同期ノイズの選択

ノイズといっても、雑音とは違います。
主に効果音に使われます。

＜周 波 数＞

チャンネル１～３の時は、周波数（Ｈｚ）を書きます。
チャンネル指定が、４又は５のときは、

０～２：三段階の決定されている周波数になります。
３　：チャンネル３で指定する周波数になります。

＜ 音 量 ＞

０　：音を消す。
１　：最小の音量。
〜
１５：最大の音量。

これによって、ゲーム等の効果音や、次表に依りメロディを奏でることができます。

周 波 数 表

音階			f1	f2	f3	f4	f5
C	ド	ハ		131	262	523	1047
C♯，D♭				139	277	554	1109
D	レ	ニ		147	294	587	1175
D♯，E♭				156	311	622	1245
E	ミ	ホ		165	330	659	1319
F	ファ	ヘ		175	349	698	1397
F♯，G♭				185	370	740	1480
G	ソ	ト		196	392	784	1568
G♯，A♭				208	415	831	1661
A	ラ	イ	110	220	440	880	1760
A♯，B♭			117	233	466	932	
B	シ	ロ	123	247	494	988	

周波数単位． Hz.

ＢＥＥＰ

プログラムの中で、短い音を出したいときに使います。

ＢＥＥＰ	ピッと音が鳴る。
ＢＥＥＰ０	ＢＥＥＰ音をとめる。
ＢＥＥＰ１	ピーと鳴り続ける。
ＢＥＥＰ２	ピポピポと鳴る。

例

```
10 A$="SEGA PERSONAL COMPUTER"
20 FOR I=1 TO LEN(A$)-1
30 PRINT MID$(A$,I,1);
40 BEEP
50 FOR J=0 TO 100:NEXT J,I
60 END
```

付　　録

変数・配列

数値変数 …… A，B，…… Z，AA，AB，…… ZZ
A0，A1 …… A9

数値配列 …… （添字，……） 3次元まで　A（15），B（5，5），
AC（3，3，3）

文字列変数 …… A$，AB$，A1$

文字列配列 …… （添字，……） 3次元まで　A$（15），
B$（5，5），AC$（3，3，3）

- 変数名は、先頭の1文字が英字、以降が英字または数字とする。長さは何文字でも良いが先頭の2文字で区別する。
- 変数と配列の名前が同じでも良い。

＜数値変数、配列の範囲＞

±9.999999999E－99
～
±9.999999999E－99

＜文字列変数、配列の範囲＞

文字長さ　　0 ～ 31

定　　数

＜数値定数＞

- 整数形式　　（例）　３，－２，９９９２６７６８
- 小数点形式　　（例）　０．２，．３，－５．３，８６．０
- 指数形式　　（例）　３，Ｅ９９，－６Ｅ３，０．３Ｅ＋５，
  ４Ｅ－８２
- １６進形式　　＆Ｈ［16進の値］……　０００００～ＦＦＦＦ
  （例）　＆Ｈ６４　……　１００と同じ
  ＆ＨＦＦＦＦ……　－１と同じ

＜文字列定数＞

〃で囲む、〃は〃を２つ用いて示す。

（例）　〃ＡＢＣ〃　→　文字　ＡＢＣ
〃〃　→　文字　NULL（なし）
〃〃〃〃　→　文字　〃
〃Ａ３〃〃６４〃　→　文字　Ａ３〃６４

内　　容	制　限
画面から内部に取り込まれる文字	80 文字
ラインバッファから予約語におとして、実際のテキストイメージにできる文字数	64 文字
文字列として扱うことの可能な文字数	31 文字
演算の優先順位等のレベル数	8 レベル
文字列演算のためのエリア	64 文字
FOR ～ NEXT のネスティングのレベル数	4 レベル
GOSUB, RETURN のネスティングのレベル数	4 レベル

キャラクタセット

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	`	p				ｰ	タ	ミ		火
1			!	1	A	Q	a	q			。	ア	チ	ム		水
2			"	2	B	R	b	r			「	イ	ツ	メ		木
3			#	3	C	S	c	s			」	ウ	テ	モ		金
4			$	4	D	T	d	t			、	エ	ト	ヤ		土
5			%	5	E	U	e	u			・	オ	ナ	ユ		♠
6			&	6	F	V	f	v			ヲ	カ	ニ	ヨ		♥
7			'	7	G	W	g	w			ァ	キ	ヌ	ラ		♦
8			(	8	H	X	h	x			ィ	ク	ネ	リ		♣
9			)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A			*	:	J	Z	j	z			ェ	コ	ハ	レ		
B			+	;	K	[	k	{			ォ	サ	ヒ	ロ	○	
C			,	<	L	¥	l	:			ャ	シ	フ	ワ	●	
D			−	=	M	]	m	}			ュ	ス	ヘ	ン	年	
E			.	>	N	^	n	~	↑		ョ	セ	ホ	゛	月	÷
F			/	?	O	π	o		←		ッ	ソ	マ	゜	日	

└コントロールコード

キャラクターコード

32	SP	48	0	64	ⓐ	80	P	96	`	112	p	128		144	
33	!	49	1	65	A	81	Q	97	a	113	q	129		145	
34	"	50	2	66	B	82	R	98	b	114	r	130		146	
35	#	51	3	67	C	83	S	99	c	115	s	131		147	
36	$	52	4	68	D	84	T	100	d	116	t	132		148	
37	%	53	5	69	E	85	U	101	e	117	u	133		149	
38	&	54	6	70	F	86	V	102	f	118	v	134		150	
39	▼	55	7	71	G	87	W	103	g	119	w	135		151	
40	(	56	8	72	H	88	X	104	h	120	x	136		152	
41	)	57	9	73	I	89	Y	105	i	121	y	137		153	
42	*	58	:	74	J	90	Z	106	j	122	z	138		154	
43	+	59	;	75	K	91	[	107	k	123	{	139		155	
44	,	60	<	76	L	92	¥	108	l	124	:	140		156	
45	−	61	=	77	M	93	]	109	m	125	}	141		157	
46	.	62	>	78	N	94	^	110	n	126	~	142		158	
47	/	63	?	79	O	95	π	111	o	127		143		159	

160		176	ー	192	タ	208	ミ	224		240	火
161	。	177	ア	193	チ	209	ム	225		241	水
162	「	178	イ	194	ツ	210	メ	226		242	木
163	」	179	ウ	195	テ	211	モ	227		243	金
164	、	180	エ	196	ト	212	ヤ	228		244	土
165	・	181	オ	197	ナ	213	ユ	229		245	♠
166	ヲ	182	カ	198	ニ	214	ヨ	230		246	♥
167	ァ	183	キ	199	ヌ	215	ラ	231		247	♦
168	ィ	184	ク	200	ネ	216	リ	232		248	♣
169	ゥ	185	ケ	201	ノ	217	ル	233		249	
170	ェ	186	コ	202	ハ	218	レ	234		250	
171	ォ	187	サ	203	ヒ	219	ロ	235		251	
172	ャ	188	シ	204	フ	220	ワ	236		252	
173	ュ	189	ス	205	ヘ	221	ン	237	年	253	
174	ョ	190	セ	206	ホ	222	゛	238	月	254	
175	ッ	191	ソ	207	マ	223	゜	239	日	255	

コマンド、ステートメント、関数

コマンド

No.	コマンド	機　　能
1	LIST	プログラムを画面に表示する。
2	LLIST	プログラムをプリンタに印字する
3	SAVE	プログラムをカセットテープに録音する。
4	VERIFY	メモリ内のプログラムとカセットテープに録音されたプログラムの比較。
5	LOAD	カセットテープのプログラムをメモリにロードする。
6	RUN	プログラムを実行する。
7	CONT	中断されていたプログラムを続行する。
8	NEW	変数、プログラムをクリアする。
9	DELETE	プログラムを部分的にクリアする。
10	AUTO	ラインナンバを自動発生する。
11	RENUM	ラインナンバをつけなおす。

NO.	ステートメント	機　　能
1	REM	コメント
2	STOP	プログラムの一時中断、CONT により続行 。
3	END	プログラムの実行終了。
4	LET	代入　（省略可）。
5	PRINT 又は ?	ディスプレイに表示する。
6	LPRINT 又は L ?	プリンターに印字する。
7	INPUT	キー入力
8	READ	“DATA” ステートメントのデータを読みとる。
9	DATA	“READ” ステートメントにより読み込まれるデータを示す。
10	RESTORE	“READ” ステートメントにより読み込む “DATA” ステートメントの場所を指定する。
11	DIM	配列の宣言。
12	ERASE	宣言された配列をクリアする。
13	DEFFN	ユーザー関数を定義する。
14	GOTO	分岐。
15	GOSUB	サブルーチンへの分岐。
16	RETURN	サブルーチンからの戻り。
17	ON ～ GOTO ON ～ GOSUB	分岐する行番号を選択する。 分岐する行番号を選択する。
18	FOR ～ TO STEP ～	NEXT ステートメントとの間を指定された条件で繰り返す。 STEP は省略化。
19	NEXT	“FOR” ステートメントによる繰り返しの場所を指定。
20	IF ～ THEN	条件分岐。
21	CONSOLE	画面スクロールの範囲、クリック音、キャラクターセットを指定する。

	NO.	ステートメント	機　　能
	22	CLS	画面をクリアする。
	23	SCREEN	画面の切替え。
	24	COLOR	色の指定
	25	PATTERN	文字、スプライトのパターンを変更する。
	26	CURSOR	表示位置の指定。
	27	POSITION	座標の指定。
	28	PSET	点をうつ。
	29	PRESET	点で消す。
	30	LINE	線を描く。
	31	BLINE	線で消す。
*	32	CIRCLE	円を描く。
*	33	BCIRCLE	円で消す。
	34	PAINT	かこまれた範囲をぬりつぶす。
*	35	SPRITE	スプライトの位置、色、パターンを指定。
*	36	MAG	スプライトの大きさを指定する。
	37	SOUND	効果音を発生させる。
	38	BEEP	BEEP 音を発生させる。
	39	HCOPY	テキスト画面の内容をプリンタにコピーする。
*	40	CALL	機械語　サブルーチンへ分岐する。
*	41	POKE	メモリーへの書き込み。
*	42	OUT	出力ポートに出力する。
*	43	VPOKE	VRAM にデータを書き込む。

＊印のステートメントは BASIC レベル 2 では使えません。

関　　数

NO.	関　　数	内　　容
1	ABS(x)	X の絶対値を求める。
2	RND(x)	乱数を作る。
3	SIN(x)	x のサインを求める。
4	COS(x)	x のコサインを求める。
5	TAN(x)	x のタンジェントを求める。
6	ASN(x)	x のアークサインを求める。
7	ACS(x)	x のアークコサインを求める。
8	ATN(x)	x のアークタンジェントを求める。
9	LOG (x)	x の自然対数を求める。
10	LGT (x)	x の常用対数を求める。
11	LTW(x)	x の 2 を底とする対数を求める。
12	EXP(x)	e の x 乗を求める。
13	RAD(x)	度をラジアン単位に変換する。
14	DEG(x)	ラジアンを度単位に変換する。
15	PI	円周率を求める。
16	SQR(x)	x の平方根を求める。
17	INT(x)	x を越えない最大の整数を求める。
18	SGN(x)	x の正負符号を与える。
19	ASC(s)	文字列 s の最初のコードを数値で与える。
20	LEN(s)	文字列 s が何文字あるかを与える。
21	VAL(s)	文字列 s を数値に変換する。

	NO.	関　数	内　　容
	22	CHR$ (x)	x の対応する文字や機能を与える。
	23	HEX$ (x)	x の 16 進数文字列を与える。
	24	INKEY$ (x)	キーボードが押されたかどうか調べる。キーボードが押されればその文字を与える。押されなければ 0 (NULL)
	25	LEFT$ (s, x)	文字列 s の左から x 番目までの文字列を与える。
	26	RIGHT$ (s, x)	文字列 s の右から x 番目までの文字列を与える。
	27	MID$ (s, x , y)	文字列 s の右から x 番目から長さ y の文字列を与える。y は省略可です。その時は最後まで与える。
	28	STR$ (x)	x をそれを表示する文字列に変換する。
	29	TIME$	内部クロックの時刻を与える。
*	30	PEEK(x)	メモリー x 番地の内容を与える。
*	31	INP(x)	入力ポートの入力内容を与える。
	32	FRE	ユーザー用メモリーエリアの空きを与える。
	33	SPC(x)	プリントステートメントで使用、スペースをあける。
	34	TAB(x)	プリントステートメントで使用、表示位置の指定。
*	35	STICK(n)	ジョイスティック n の方向を示す。
*	36	STRIG (n)	ジョイスティック n のトリッガーボタンの状態を示す。
*	37	VPEEK(x)	VRAM x 番地の内容を与える。

エラーメッセージ

1 表 示 FORMAT

(1) コマンド入力又はステートメントをダイレクトに入力した時、エラーが発生した場合。

? メッセージ error

(2) テキストを実行中にエラーが発生した場合。

? メッセージ error in ラインナンバー

(3) **INPUT** ステートメントによるキー入力データに誤りがあった場合。

? メッセージ

メッセージ	内　　容
System	BASICインタプリンタのプログラムの動作エラー。 （普通はありえない）
N−formura too Complex	数値式が複雑すぎる。
S−formura too Complex	文字列式が複雑すぎる。
Overflow	値や演算結果が許容範囲を越えた。
Division by Zero	除算の分母が 0 。
Function Parameter	関数のパラメタが異常。
String too long	文字列の長さが 64 をオーバーした。
Stack overflow	・カッコの使いすぎ。　・PAINT する図形が複雑すぎる。 ・ユーザ定義関数が自分自身を呼び出している。
Out of memory	メモリが足りない。　・テキスト　・変数　・配列
Number of Subscripts	添字の数（個数）が異常。
Valve of Subscript	添字の値がおかしい。
Syntax	文法上のエラー。
Command Parameter	コマンドのパラメタが異常。
Line number over	AUTO 又は RENUM においてラインナンバーが 65535 オーバーする。
Illegal line number	ラインナンバーがおかしい。
Line image too long	ラインが長くなりすぎる。（ RENUM 等）

メッセージ	内　　容
Undefined line number	指定されたライン番号がない。（**RENUM , GOTO , GOSUB , IF–THEN , RESTORE , RUN** ）
Type missmatch	代入する側と代入される側のタイプが合わない。（数値、文字列）
Out of DATA	**READ** ステートメントにより読み込もうとしたが、**DATA** 文のデータがない。
RETURN without GOSUB	**GOSUB** を行なわないで **RETURN** 文が実行された。
GOSUB nesting	**GOSUB** の入れ子が 4 レベルをこえた。
NEXT without FOR	**NEXT** に対応する **FOR** ステートメントが無い。
FOR nesting	**FOR ～ NEXT** の入れ子が 4 レベルをこえた。
Statement Parameter	ステートメントのパラメタが異常。
Con't continue	**CONT** コントによる続行が不可能。
FOR variable name	**FOR** ステートメントのループ変数が数値変数でない。（文字型又は配列）
Array name	**DIM** ステートメントのパラメタが配列でない。
Redimensioned array	配列を 2 重に定義しようとした。
Undefined array	未定義の配列を **ERASE** しようとした。
No program	プログラムがテキスト中に無いのに **SAVE** しようとした。
Memory writing	メモリへの書き込みエラー。（ **LOAD** 時）

メッセージ	内　　容
Device not ready	プリンタが接続されてない、又は故障。
Undefined function	定義されてないユーザ関数を呼び出した。
Verifying	テープのプログラムとの比較エラー。
Illegal direct	ダイレクトステートメント実行不可。
Redo from start	**INPUT** ステートメント入力データがおかしい。 ・最初から入力しなおしを要求。
Extra ignored	**INPUT** ステートメント入力データがおかしい。 ・余分なデータが入力された。　・余分なデータは無視。
Unprintable	上記以外のエラー。

SC-3000 BASIC レベル 2 解説書

発行所 株式会社 セガエンタープライゼス

パーソナルコンピュータ事業部

〒144 東京都大田区羽田1丁目2番12号

☎ 03 − 742 − 3171

昭和58年7月15日発行

印刷・製本 (有) 日 本 現 図 社

〒140 東京都品川区北品川1丁目29番11号

☎ 03 − 471 − 3353

SEGA

セガ・エンタープライゼス